Department of
Computer Science

Technical Report

Undergraduate Dissertation: Extracting Semantics from the Enron Corpus

Tim Macfarlane

Technical Report 2013-08 **November 2013**

ISSN 1740-9497

Contact Address:
Department of Computer Science
University of Bath
Bath, BA2 7AY
United Kingdom
URL: http://www.cs.bath.ac.uk

ISSN 1740-9497

Extracting Semantics from the Enron Corpus

Tim Macfarlane

Bachelor of Science in Computer Science with Honours
The University of Bath
May 2013

Extracting Semantics from the Enron Corpus

Submitted by: Tim Macfarlane

COPYRIGHT

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed:

Abstract

Indirect measures must be used when analysing attitudes, as individuals are unlikely to voluntarily express beliefs that are opposed by social norms. The IAT indirectly assesses attitudes through the automatic association of concepts and attributes, however it requires strict control of extraneous influences. This paper proposes an alternative indirect measure of attitudes by designing a semantic space of the way in which words are used in language. To demonstrate the use of semantic spaces, the Enron corpus is analysed to discover whether any cultural attitudes can be observed. In the preprocessing stage, the corpus is tokenised, lemmatised and irrelevant information to semantic analysis is removed. The Enron Semantic Space is then created from the corpus, incorporating multiple features from Hyperspace Analogue to Language (HAL), Latent Semantic Analysis (LSA) and Lowe and McDonald's Semantic Space (LMS). A free association test is then introduced to analyse the accuracy that the system can observe direct cognitive priming. Features from LMS and LSA are selected over HAL in the optimum implementation as they give the best accuracy of 86.86% on the free association test. The same features are also shown to be able to observe graded and mediated priming. After, an application is presented that allows a user to create an Enron Semantic Space from scratch, and compare the differences in similarities of concepts and attributes found in the space. Using this application a numerous amount of attitude experiments are conducted. *Life* words are found to be associated to *pleasant* words and *death* words associated to *unpleasant* words. *Enron* is also found to be more similar to *pleasant* words than *Dynergy*. *Competence* words are found to be associated with *youth* words and *incompetence* words associated with *elderly* words. Furthermore, *career* words are found to be associated with *male* words and *family* words with *female* words. Finally, we conclude that the results support the argument towards using a semantic space to analyse attitudes, however supplementary studies need to be conducted to replicate exact experiments conducted by the IAT.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor, Joanna Bryson, whose has provided invaluable advise and support throughout the duration of this project.

Chapter 1

Introduction

In psychology, it is assumed that personal attitudes and beliefs play an extremely important role in social cognition (Fishbein and Ajzen, 2005). Attitudes are an individual's positive or negative evaluation towards a specific entity (Ajzen and Fishbein, 1977). Subsequently, it is considered that these positive and negative feelings are consistent with behaviour; for example if we view a person in a positive light we are expected to act in a favourable way towards them rather than in a unfavourable way. Landy and Sigall (1974) discovered that essays written by female students were given a higher grade by male participants when presented with a photo that showed the author to be physically attractive, compared to a photo showing the author to be physically unattractive. Due to this relationship between behaviour and attitudes, researchers have continually tried to use attitudes as a predictive tool for many different types of behaviours. Examples of these studies include endeavouring to predict consumer behaviour (Danziger, 2004), eating behaviour (Hofmann, Rauch and Gawronski, 2007), voting behaviour (Arcuri, Castelli, Galdi, Zogmaister and Amadori, 2008) and suicidal behaviour (Nock, Park, Finn, Deliberto, Dour and Banaji, 2010).

The main obstacle encountered when trying to assess the predictive quality of attitudes is the method of observing attitudes itself. One approach in attempting to observe attitudes is using a *direct measure* such as *self reporting*, where the user responds to a series of questions concerning their views towards a particular subject. A major flaw with this method is that the responses could largely be influenced by what the participants perceive to be socially acceptable rather than their actual beliefs (Fazio and Olson, 2003). Therefore, if a person has an attitude that is not socially acceptable then they are unlikely express it in an interview. In fact, Constantine and Ladany (2000) found that there was a significant correlation towards self reporting measures and indexes of general social desirability. A solution to this problem is to use *indirect measures* rather than *direct measures* to observe attitudes.

1.1 Indirect Measures of Attitudes

Indirect measures of attitudes are those where:

- The participant is unaware that an attitude is being measured
- The participant has no control over the outcome of the experiment

(De Houwer, 2006)

These measures are based on the theory that attitudes can manifest from automatic cognitive evaluation that one has no control over (Greenwald and Banaji, 1995). In order to observe this 'automatic cognition', indirect measures restrict the mental capacity of the participant and require instinctive responses (Hintzman, 1990; Merikle and Reingold, 1991). In these experiments the participant is unable to explicitly identify that an attitude is being measured and therefore is unable to hide the socially undesirable attitudes that can be masked during self reporting.

The most recent example of an indirect measure is the Implicit Association Test (IAT) (Greenwald et al., 1988), which measures the automatic associative strength between concepts in memory. This test has found many interesting results including consistent racial attitudes, gender attitudes and age attitudes across a range of participants. However, as mentioned in Chapter 2, there are many extraneous influences which can affect the reliability results obtained using an IAT and are rather complicated to control for (Nosek, Greenwald and Banaji, 2007).

In this project a different attitude measure is proposed, through the analysis of the way in which people use words in written language. This is largely influenced by the research conducted by Bilovich (2006) and Bilovich and Bryson (2008).

1.2 Attitudes Expressed in Natural Language

Language is an extremely useful tool that we use everyday to interact and communicate with our natural environment. Freud (1938) noted that the way we communicate and the words we use give an insight into our physical, mental and emotional state. Therefore, natural language can be seen as a behavioural medium through which we can observe cognitive processes (Lacan and Wilden, 1968; Ricoeur, 1976). When we use a word in a positive context, this could give an indication that we view the word in a positive light, and if two words are used in a similar way this could be an indication that we believe that these two words are associated. Consequently, if one had a sizeable collection of an individual's word use then this could be utilitised to extract their attitudes.

Bilovich (2006) adopted this idea by creating a statistical model of word use in the British National Corpus called a *semantic space*. Using this semantic space, Bilovich (2006) observed near universally held attitudes towards flowers, insects, instruments and weapons, also found in Greenwald et al. (1988)'s IAT.

1.3 Project Objective: Analysing Cultural Attitudes in the Enron Corpus

This project attempts to supplement the work done by Bilovich (2006) and Bilovich and Bryson (2008), by creating a *semantic space* of word use in the emails of the Enron Corporation, contained in the Enron Corpus. Using this semantic space we will endeavour to extract the cultural attitudes expressed by the people who were employed by Enron. In doing so, it may be possible to replicate similar findings to the IAT and further support the use of semantic spaces as an indirect measure of attitudes.

Before we describe the processes of creating a semantic space of Enron emails we first introduce the background research that influenced the design process. Chapter 2 describes the Implicit Association Test and how it observes attitudes through use of cognitive priming. In Chapter 3 we introduce the different types of cognitive priming and the non-statistical approaches which attempt to model them. Finally, in Chapter 4 we describe semantic spaces as an alternative approach in modelling cognitive priming and thus an appropriate choice for observing attitudes in the same way as the IAT.

Chapter 2

Implicit Association Test

The Implicit Association Test (IAT) was designed by Greenwald et al. (1988) after being greatly influenced by work on affective priming (Fazio, Jackson, Dunton and Williams, 1995). The experiment selects two concepts such as *male* and *female* gender and two attributes such as *mathematics* and *art*. Its primary goal is to observe whether the participant has a tendency to associate a concept with one of the attributes and the other concept with the other attribute. In the *male/female*, *mathematics/art* example the test observes whether the participant holds the general stereotype that men are more associated with studying mathematics and females are more associated with studying art. As well as observing stereotypes, this experiment can also be extended to attitudes; for example the two attributes could represent positive and negative descriptions such as *pleasantness* and *unpleasantness*. If a concept is associated with the positive attribute, we can hypothesise that the participant has a positive attitude towards the concept. Conversely, if the participant associates a concept with a negative attribute then the participant has a negative attitude towards that concept. In the first section of this chapter, the procedural steps taken by the IAT are described, using the experiment conducted by Greenwald et al. (1988) as an example. A summary is then provided of current empirical findings which support the IAT before introducing the extraneous influences that need to be strictly controlled for in the experimental procedure.

2.1 Experimental Procedure

In their original paper, Greenwald et al. (1988) designed a test to study the difference between African American and White American racial groups and their association with pleasantness. The experiment used 4 types of stimuli:

- African American male and female names
- White American male and female names

- Pleasant associated words
- Unpleasant associated words

In the test the concept titles **Black** and **White** and the attribute titles **Pleasant** and **Unpleasant** were placed in the top corners of a computer screen. When presented with a stimulus the participant was told to press the left key if the stimulus was associated with the concept/attribute in the top left hand corner of the screen and the right key if the stimulus was associated with the concept/attribute in the right hand corner of the screen. The original experiment progressed in 5 stages:

1. **Black** was placed to the left, **White** to the right. Participants were asked to categorise names.
2. **Pleasant** was placed to the left, **Unpleasant** to the right. Participants were asked to categorise the unpleasant and pleasant words.
3. **Black** and **Pleasant** were placed to the right, **White** and **Unpleasant** to the left. All stimulus was used and reaction times for each individual stimulus was measured.
4. **White** was placed to the left, **Black** to the right. Participants were asked to categorise names.
5. **White** and **Pleasant** were placed to the left, **Black** and **Unpleasant** to the right. All stimulus was used and reaction times for each individual stimulus was measured.

The first two stages allowed the participant to learn the correct response to the stimulus. If a stimulus was associated with the wrong category then an error would appear notifying the user that the stimulus was associated with the other category. In the third stage, participants' responses to the pairing of *black/pleasant* and *white/unpleasant* were timed. In order to reverse the pairings, stage 4, then re-trained the participant to respond with right for *black* names (instead of left previously), and left for *white* names. The final stage then timed the responses to the pairings *white/pleasant* and *black/unpleasant*. Figure 2.1 illustrates this process.

The theory behind this experiment is that if the concept/attribute pair is strongly associated then it will produce a faster response time than a concept/attribute pair that is weakly associated. So, if the results produce faster responses for the pairing *white/pleasant* and *black/unpleasant*, than the pairing *black/pleasant* and *white/unpleasant*, then we can assume that the individual automatically associates White Americans as being more pleasant than African Americans. Subsequently, they have a positive attitude towards White Americans and a negative attitude towards African Americans.

Greenwald et al. (1988) validated results from the IAT by performing a test to ascertain whether universally accepted attitudes could be demonstrated. *Flowers* were compared

Sequence	1	2	3	4	5
Task description	*Initial target-concept discrimination*	*Associated attribute discrimination*	*Initial combined task*	*Reversed target-concept discrimination*	*Reversed combined task*
Task instructions	● BLACK WHITE ●	● pleasant unpleasant ●	● BLACK ● pleasant WHITE ● unpleasant ●	BLACK ● ● WHITE	BLACK ● ● pleasant ● WHITE unpleasant ●
Sample stimuli	MEREDITH ○ ○ LATONYA ○ SHAVONN HEATHER ○ ○ TASHIKA KATIE ○ BETSY ○ ○ EBONY	○ lucky ○ honor poison ○ grief ○ ○ gift disaster ○ ○ happy hatred ○	○ JASMINE ○ pleasure PEGGY ○ evil ○ COLLEEN ○ ○ miracle ○ TEMEKA bomb ○	○ COURTNEY ○ STEPHANIE SHEREEN ○ ○ SUE-ELLEN TIA ○ SHARISE ○ ○ MEGAN NICHELLE ○	○ peace LATISHA ○ filth ○ ○ LAUREN ○ rainbow SHANISE ○ accident ○ ○ NANCY

Figure 2.1: Sequence model from Greenwald et al. (1988).The black dots represent the side that the category was placed and the white dots represent the correct stimulus response.

against *insects* and *musical instruments* against *weapons* with their associations to pleasantness. The results supported their hypothesis as flowers and musical instruments were more associated with pleasant words and insects and weapons were more associated with negative words.

2.2 Empirical Findings Using the IAT

Since its creation in 1998, the IAT has gained a large amount of support due to its advantage over self reports (Fazio and Olson, 2003). Figure 2.1 summaries a few of the findings that have been published using the IAT.

Greenwald et al. (1988)	Studied within group racial bias. They found that Japanese and Korean students associated surnames used in their own culture with being more pleasant than surnames used in the other culture. They also found that their was a significant racial bias (White American vs African American, pleasant vs unpleasant) in white college students.

Rudman, Greenwald and McGhee (2001)	Studied *gender* vs *potency* and *gender* vs *warmth* between males and females. They found a strong bias for male candidates to associate men with strength (potency) more than women with strength. They also found that participants on average viewed their gender as being warmer/kinder than the opposite gender.
Nosek, Banaji and Greenwald (2002)	Studied *gender* with *career* vs *family* and found that there was a significant bias in associating men with career and women with family.
Hummert, Garstka, O Brien, Greenwald, Mellott et al. (2002)	Studied *age* vs *pleasantness* within people of all ages and found that there was a significant *youth/pleasantness* and *old/unpleasantness* bias.
Maison, Greenwald and Bruin (2004)	Managed to successfully predict consumer preference between types of yoghurts, soft drinks and fast food restaurants.
Arcuri et al. (2008)	Studied political attitudes of undecided voters before an election and found that implicit biases correlated strongly with their actual voting behaviour.

Table 2.1: Summary of some empirical findings using the IAT

2.3 Extraneous Influences

Nosek et al. (2007) outline 4 major extraneous factors that can harm the validity of results obtained by the IAT:

Order of Combined Tasks: Greenwald, Nosek et al. (2001) found that the first position of concepts in step 3 interfered with the task where the positions of concepts were switched in step 5. Even though participants were given some practice to respond with the alternate key, the participants still had a delayed response where they tried to resist the temptation to respond with the key learned in the first task. Due to this, there was a tendency to find that participants responded more slowly to step 5 because of this interference, rather than a weaker association with concepts. Although increasing the number of training responses required in step 4 reduces this factor, there could still be a tendency for the first button/concept pairing to interfere with the second button/concept pairing.

Cognitive Fluency: Nosek et al. (2007) describe *cognitive fluency* as the average response latency for a participant. Greenwald, Nosek, Banaji et al. (2003), found that slower participants on the whole tended to have larger IAT effect than participants who

responded quickly. In order to cope with this affect they created a scoring algorithm that significantly reduces this effect.

Subject Age: Greenwald et al. (2001) discovered younger participants had a better cognitive fluency than older participants and therefore older participants had a tendency to have larger a IAT effect than younger participants.

Previous Experience with IAT: Greenwald et al. (2001) also found that participants who had previous experience with the IAT had faster reaction times than participants with no previous experience. They found that participants who had previous experience were less like to have IAT effects. Both this effect and the subject age effect in the previous point can be controlled for by using the same scoring algorithm used by Greenwald et al. (2003). However, even though these effects are significantly reduced they still remain.

2.4 Chapter Discussion

In this chapter we have discussed the IAT test as an indirect measure of attitudes through automatic associations of concepts and attitudes. Even though it has gained a large amount of support and empirical findings, there are still a range of extraneous influences that can dramatically affect the reliability of the IAT effects observed. Subsequently, we hypothesise that the same automatic association of concepts and attitudes can be observed by analysing a semantic space of word use. Because this indirect measure requires no participants to be involved (other than providing a sample of their language use), none of the extraneous effects mentioned in the section above will be present, therefore the results we obtain could be more reliable. Before we introduce the semantic space approach, in next chapter we discuss the reason why we can assume that faster responses towards a concept/attribute pairing means that they are strongly associated. This evaluation can be made in confidence because of the phenomenon called *cognitive priming* (Greenwald et al., 1988).

Chapter 3

Cognitive Priming

Cognitive priming is the effect where recall of a concept in memory can be facilitated by the presentation of an associated word and inhibited by the presentation of an unassociated word. This chapter discusses the three major types of cognitive priming and two non-statistical models of memory that attempt to account for it.

3.1 Types of Cognitive Priming

There are three major types of cognitive priming: *direct*, *graded* and *mediated*. Each one will be discussed in turn.

3.1.1 Direct Cognitive Priming

Direct cognitive priming is the simplest form of word priming. It is the theory that the recognition of a target word can be facilitated by the presentation of a directly associated word called a *prime*. For example, the word *cup* is a prime to the target word *coffee*, and when the prime is presented before the target, the target will be recognised faster than if an unrelated word was presented before it such as *dog*.

Meyer, Schvaneveldt et al. (1971), were first noted as finding this phenomenon; their study involved a recognition task (or lexical decision task), where the participants were presented with a string of letters and had to respond with *yes* if the string of letters was a word and *no* if the string of letters was a non-word. They found that the words were recognised faster when they were preceded by a prime than when they were preceded by a unrelated word. This priming effect has been found to occur when both target and prime are visually presented (Meyer et al., 1971; Frost, Forster and Deutsch, 1997), both aurally presented (Marslen-Wilson, Tyler et al., 1997; Marslen-Wilson and Zhou, 1999) and when the prime is aurally presented and the target is visually presented (Marslen-Wilson, Komisar-

jevsky Tyler, Waksler and Older, 1994). Balota and Lorch (1986), also found a significant priming affect when the participant was asked to pronounce the target word rather than decide its validity.

The two major extensions to direct cognitive priming are *graded* and *mediated* priming.

3.1.2 Graded Priming

Graded priming is where strongly associated words will provide better priming affects than weakly associated words. McKoon et al. (1992) proposed that the degree to which two words are associated affect the speed of the recognition. In doing so, they created a hierarchy of association:

$$\text{Free Associate} > \text{High T-Prime} > \text{Low T-Prime} > \text{Unrelated}$$

Free associates are words that are generated with high probability in an free association experiment. In this experiment participants are given a target word and are asked to respond with the first word that comes to mind. If *cat* is generated with a high probability in response to the target word *dog* then it is said to be a free associate of *dog*.

T-primes are words that co-occur with the target word more often than chance (Church and Hanks, 1990). To calculate this, a statistical method called Pointwise Mutual Information (Fano, 1961; Palermo and Jenkins, 1964) is used:

$$tscore(x, y) = log(\frac{p(x, y)}{p(x)p(x)}) \tag{3.1}$$

The ratio measures the probability of observing the word x and word y together over the probability of observing them independently. If x and y occur more often than chance then their joint probability $p(x, y)$ will be greater than the product of their independent probabilities $p(x)p(y)$ and resulting in a t-score greater than 0. Using the Associated Press Newswire Corpus of 6 million words, Church and Hanks (1990) estimated $tscore(x, y)$ by counting the number of times that x was followed by y in a window of 6 consecutive words. They then discarded any words which gave a negative $tscore$. The *high t-prime* words that were used by McKoon et al. (1992) were primes that gave a high $tscore$ and similarly *low t-prime words* were those that gave a low $tscore$.

In their study McKoon et al. (1992) gave 52 psychology students a lexical decision task similar to Meyer et al. (1971). There were 4 experimental conditions, one where the cue was a free association norm, one where the cue was a high t-prime, one where the cue was a low t-prime and one where the cue was an unrelated word (chosen from the prime of another word). The results supported their hierarchy of association as free-associates provided the quickest response times followed by high t-primes, low t-primes and then unrelated words.

3.1.3 Mediated Priming

Mediated priming occurs when a triad forms between three words (Balota and Lorch, 1986). For example if A is related to B, and B is related to C, then A is also weakly associated to C; we say that A and C are mediated primes. For example, *tiger* is related to *lion* and *stripes* is related to *tiger*, therefore *stripes* is a mediated prime of *lion*. Subsequently, a similar (although smaller) increase in recognition can occur when a mediated prime precedes a target word. Balota and Lorch (1986), observed this effect in a pronunciation task and McNamara and Altarriba (1988), in a lexical decision task. In both cases, they found that mediated primes produced faster recognition times than unrelated words and direct primes produced faster recognition times than mediated primes.

3.2 Non-statistical Models of Semantic Memory

Now that we have looked at the different forms of cognitive priming, we investigate two non-statistical models of semantic memory and how they explain this phenomenon. These models are *spreading activation theory* and *compound cue theory*.

3.2.1 Spreading Activation Theory

In spreading activation theory, knowledge in memory is described as being an undirected graph or semantic network (see Figure 3.1; Collins and Loftus (1975); Anderson (1983)). In this network nodes represent concepts and are called *cognitive units*. Each cognitive unit is connected to other cognitive units through associative links or edges. Examples of some types of associative links are membership of the same category (e.g. *car* and *truck* are both vehicles), sub-ordinate/super-ordinate links (e.g *car* is a *vehicle*) and contextual links (e.g. *street* and *car* are contextually similar) (Collins and Loftus, 1975; Quillian, 1967).
When a word is loaded into the semantic network, activation spreads from the node to its neighbouring nodes and this continues recursively to the neighbours of its neighbouring nodes and so on (Anderson, 1983). This explains the direct priming phenomenon as activation spreads from the prime in one direct stage to its neighbouring node, the target word. The strength of the association between two nodes is the aggregate of all their overlapping properties (Collins and Loftus, 1975). For example, this means that membership is stronger than descriptive links (i.e in Figure 3.1 *fire-engine* is nearer to *car* than it is to the colour *red*). Stronger associations have shorter pathways between nodes which accounts for graded priming as less associative primes have longer pathways and therefore take longer to activate. Finally, the activation time of mediated primes is the sum of the <*target*,*prime*> edge and <*prime*,*mediated prime*> edge (Balota and Lorch, 1986).

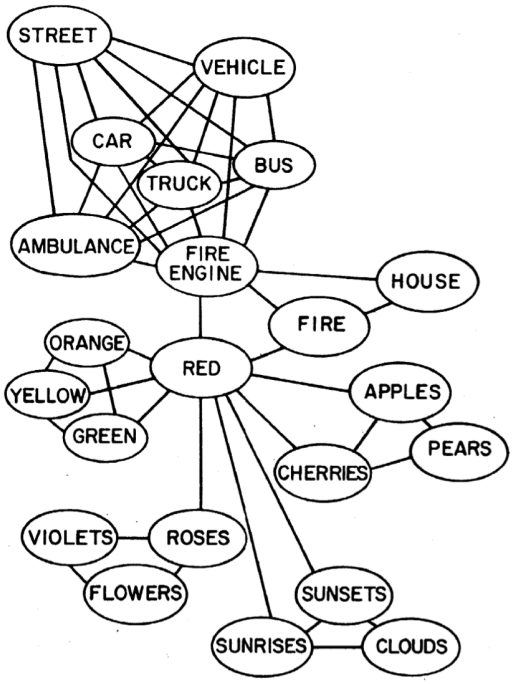

Figure 3.1: Example of organisation of concepts in memory taken from Collins and Loftus (1975). Shorter length of edges between nodes represent stronger associative strength.

3.2.2 Compound Cue Theory

In compound cue theory, semantic memory is arranged as a set of images or episodes (Ratcliff, McKoon et al., 1988; Dosher and Rosedale, 1989). When a set of stimulus is presented to short-term memory each cue within that set is formed into a compound cue. During the retrieval process, the compound cue is compared with all images in memory and a familiarity value is calculated. The higher the familiarity value, the faster the retrieval.

The calculation of familiarity can be explained using the Search Associative model of Memory (SAM) (Gillund, Shiffrin et al., 1984). In a simple example Figure 3.2 shows the association matrix of numbers as described by SAM.

In SAM each cue in the compound is encoded separately, compared with an image and the strength of the association is found; which is called $S_{c,j}$. Figure 3.2 shows the strength values for numbers. In this matrix, cues paired with images equal to themselves are given a strength of 1 (i.e $S_{1,1}$, $S_{2,2}$, etc). Cue-image pairs that have a direct link also have a strength value of 1 (i.e. adjacent numbers; $S_{1,2}$, $S_{2,3}$, etc), whereas non-related cue/image pairs are given the value of 0.2. The strength of association of a compound cue with an image is the product of the strengths of each cue and that image. The familiarity of a cue

	Target									
Cue	1	2	3	4	5	6	7	8	9	10
1	1	1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
2	1	1	1	0.2	0.2	0.2	0.2	0.2	0.2	0.2
3	0.2	1	1	1	0.2	0.2	0.2	0.2	0.2	0.2
4	0.2	0.2	1	1	1	0.2	0.2	0.2	0.2	0.2
5	0.2	0.2	0.2	1	1	1	0.2	0.2	0.2	0.2
6	0.2	0.2	0.2	0.2	1	1	1	0.2	0.2	0.2
7	0.2	0.2	0.2	0.2	0.2	1	1	1	0.2	0.2
8	0.2	0.2	0.2	0.2	0.2	0.2	1	1	1	0.2
9	0.2	0.2	0.2	0.2	0.2	0.2	0.2	1	1	1
10	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	1	1

Figure 3.2: Numerical example of semantic memory modelled by SAM taken from Ratcliff et al. (1994). The values in the table represent the strength of the relationship between the cue and target image, $S_{c,i}$

.

is therefore the sum over its associative strength for all the images. An example of the familiarity of a compound cue that contains a prime p and a target word t is taken from Ratcliff et al. (1988):

$$F(p,t) = \sum_i S_{p,i} S_{t,i} \tag{3.2}$$

Therefore the familiarity of a compound cue containing the prime *1* and target *2* is calculated as:

$$F(1,2) = S_{1,1} * S_{2,1} + S_{1,2} * S_{2,2} + S_{1,3} * S_{2,3} ... = 1.0 * 1.0 + 1.0 * 1.0 + 0.2 * 1.0 ... = 2.48 \tag{3.3}$$

This model supports direct priming as the in example for *F(1,2)*. This will produce high strength values for images *1* and *2* because he are directly related. It also supports mediated priming as *1* and *3* will both share an association with the image *2*. Finally graded priming is also supported as the range of priming is defined by the number of images that the prime and target are both associated with (McKoon et al., 1992).

3.2.3 Problems with Non-statistical Models

The problem with these non-statistical models is that they are extremely difficult to implement. The main reason for this is that spreading activation and compound cue models cannot be learned and the semantic representation of memory has to be built from scratch (Jones, Kintsch and Mewhort, 2006). This obviously relies on the designer of the implementation having a good knowledge of associative strengths and the dimensionality of semantic memory. As this is knowledge fairly intractable, it is therefore unlikely that the

designer will be able to create an accurate representation (Hummel, Holyoak et al., 2003). Furthermore, it is completely impractical due to the size of semantic memory and would take an extremely long time to build.

3.3 Chapter Discussion

In this chapter we have introduced cognitive priming and its three types: direct, graded and meditated. Moreover, both graded and mediated priming can be used to explain the evaluation process of the IAT. For example, in the IAT, consider the concept *male*, the attribute *mathematics* and the concept stimulus of a *male name*. When we see the *male name*, the attribute *mathematics* acts as a mediated prime because *mathematics* is associated as being studied my *men* and *male names* are associated with the concept *men*. Therefore, there are quicker response times for the pairing *men*/*mathematics* due to mediated priming. Furthermore, quicker responses to the paring *men*/*maths* than *women*/*maths* is explained by graded priming; the fact that *men* is a stronger associate to *maths* than *women* is to *maths*.

Finally, we have presented two non-statistical modes that are used to explain how cognitive priming occurs in long term memory. Although spreading activation and compound cues both give plausible descriptions of semantic memory, neither of which are practical to implement. In order to analyse cultural attitudes in the Enron Corporation, we must find a modal that can be learned by processing the content of emails in the Enron corpus. To ensure our model replicates the automatic association of concepts and attributes in a similar way to IAT, it needs to be able to observe all types of cognitive priming. In the next chapter we will introduce a group of models which have shown the ability to do both of these. This group is called *semantic space models*.

Chapter 4

Semantic Spaces

Semantic spaces are based on the assumption that a word is defined by the context that it in which it is used (Wittgenstein, 1958). Thereby, a word can be represented as an n-dimensional vector of its weighting in all contexts. Words that are associated with one another will be used in similar contexts and will have many features that overlap. The stronger the association between the two words the more features that will overlap. Similarly, by plotting words as points in a n-dimensional space defined by their context, words that are associated will be closer to each other than words that are unassociated. Semantic space models are therefore constructed as a matrix of words and their co-occurrence with contexts. For this reason they are more practical than non-statistical models as they can be learned from a corpus of text.

This chapter describes three approaches to designing semantic spaces: Hyperspace Analogue to Language (HAL) (Lund, Burgess and Atchley, 1995; Lund and Burgess, 1996; Burgess, 1998), an adaption of the HAL model by Lowe and McDonald (2000) (which we refer to as LMS, for Lowe and McDonald's Space) and Latent Semantic Analysis (Landauer, Foltz and Laham, 1998).

4.1 Hyperspace Analogue to Language

In HAL the context of a word is defined by the words that neighbour it (Lund et al., 1995; Lund and Burgess, 1996; Burgess, 1998). Thus the semantic space defined by HAL is a word-by-word matrix of co-occurrences for all words in the corpus. The rows of the matrix are the target words themselves and the columns are a set of context words. To construct the matrix a window of fixed size is passed through the corpus counting the number of times a target word occurs after a context word. A value is also given to the strength of the co-occurrence which is inversely proportional to the number of words separating the two words. For example, in Figure 4.1 words occurring directly after the target word are given a value of 5 (the size of the window) and this value decreases by 1 as the distance between

the context word and the target word increases. The result of this is an n x n matrix of co-occurrences. Optionally the columns and rows of the same word can be concatenated to produce a 2n x 2n matrix; this combines the use of the target word before and after context words (Lund and Burgess, 1996). Finally, the matrix is normalised so that each row vector adds up to 1.

Table 1
Example Matrix for "The Horse Raced Past the Barn Fell"
(Computed for Window Width of Five Words)

	barn	fell	horse	past	raced	the
<PERIOD>	4	5	0	2	1	3
barn	0	0	2	4	3	6
fell	5	0	1	3	2	4
horse	0	0	0	0	0	5
past	0	0	4	0	5	3
raced	0	0	5	0	0	4
the	0	0	3	5	4	2

Figure 4.1: Example of a co-occurrence matrix in HAL, taken from Lund and Burgess (1996).

The size of the window defines the distance for two words considered to be co-occurring. A small word window may miss out important information such as long phrases (Lund et al., 1995), whereas a large word window may include a large amount of extraneous occurrences. Lund et al. (1995), suggest that a 10 word window is appropriate and that any extraneous occurrences happening towards the edge of that window will be mediated by the weighting applied to the strength of the co-occurrence.

The similarity between the two words can then be ascertained by calculating the distance between their row vectors. In HAL this is found by using the Euclidean distance (Lund et al., 1995; Burgess, 1998), where if $X = (x_1, x_2....x_n)$ and $Y = (y_1, y_2...y_n)$ are vectors then:

$$distance(X, Y) = \sum_i \sqrt{(x_i - y_i)^2} \tag{4.1}$$

The smaller the distance between the two word vectors the more associated they are.

4.1.1 Dimensionality Reduction in HAL

In their original study Lund et al. (1995) analysed the 70,000 most frequent words in the USENET corpus. This therefore created a 70,000 x 70,000 matrix, which is computationally expensive. To solve this issue, the variance of each context word was calculated and the columns with the least variance were removed. Lund et al. (1995) hypothesised

that the context words with the most variance contained the most important information. Subsequently, a matrix containing only the most variant context words would be a good approximation of the original matrix and reduce the computational complexity and memory consumption. Lund et al. (1995) discovered that a word vector of 200 elements was the best compromise between the content and complexity of the model.

4.2 Lowe and McDonald's Space

LMS is similar to HAL in that it also creates a word x word matrix of co-occurrences using a window technique. However, instead of weighting the occurrence of a target and context word by distance, all co-occurrences are given a value of 1. Moreover, Lowe and McDonald (2000) focus on the problem presented by Zipf's Law of language (Zipf, 1949), and why raw frequency counts should be avoided.

4.2.1 Zipf's Law and the Log-odds-ratio

Zipf's law (Zipf, 1949) emphasises the problem of data sparsity within language. It notices that the number of times a word occurs in language is inversely proportional to its rank. An alternative way to view this is if we rank words by frequency, then the rth most frequent word will occur 1/r times the frequency of the most frequent word. Therefore, few words will occur frequently within text, whereas a larger number of words will occur very infrequently, signifying that words are not normally distributed across language. If two words are unrelated, but they occur with similar frequency in language, then they will have vectors of similar magnitude and will be judged to be similar (Lowe, 2000; Lowe and McDonald, 2000). Therefore Zipf's law suggests that any information retained in the raw co-occurrence of a context word and a target word will be convoluted by the chance of occurrence of either word.

In order to constrain the affect of chance co-occurrences, Lowe and McDonald (2000) convert the frequency of the co-occurrence of a context word c and target word t into association strengths using the *log odds ratio* (Agresti, 1990):

$$log(\theta(c,t)) = log(\frac{f^W(c,t) * f^W(\neg c,\neg t)}{f^W(c,\neg t) * f^W(\neg c,t)} \tag{4.2}$$

In this equation $f^W(c,t)$ is the frequency of co-occurrences of c and t within a window of size W. Using the log odds ratio, if $\theta > 1$, then the presence of c increases the chance of seeing t, if $\theta = 0$ then both words are distributionally independent and if $\theta < 0$ then the presence of c decreases the chance of seeing t. In the model used by Lowe and McDonald (2000), all negative values are set to 0 as they are viewed as being more psychologically salient. Finally, a logarithmic function is applied to θ in order to scale large values. The advantage of this equation is that $f^W(c,\neg t)$, $f^W(\neg c,t)$ and$f^W(\neg c,\neg t)$ can all be approximated using

$f(c,t)$, the size of the window used W and the size of the corpus N:

$$f^W(c,\neg t) = Wf(c) - f^W(c,t) \tag{4.3}$$

$$f^W(\neg c,t) = Wf(t) - f^W(c,t) \tag{4.4}$$

$$f^W(\neg c,\neg t) = WN - (f^W(c,t) + f^W(c,\neg t) + f^W(\neg c,t)) \tag{4.5}$$

4.2.2 Similarity used by LMS

Instead of using the Euclidean distance like HAL, Lowe and McDonald (2000) use the Cosine angle between the two word vectors. The main reason for this is that the Cosine value ranges between -1 and 1, reducing the need for any scaling technique to be implemented. The Cosine between the two word vectors u and v is given as:

$$cos(u,v) = \frac{u.v}{||u||.||v||} \tag{4.6}$$

As the word vectors found by LMS will always be positive this value will be in the range of [0,1] where 0 is completely dissimilar and 1 is an exact similarity. Therefore the higher the Cosine value, the more associated two words are.

4.2.3 Context words used by LMS

As mentioned in section 4.1.1, HAL only includes the most variable context words in the semantic space because Lund et al. (1995) hypothesise that these context columns will contain the most important information. Lowe and McDonald (2000), take an alternative approach by choosing the most reliable words from a collection of the most frequently occurring words in the corpus. A context word is reliable if it produces similar target word co-occurrence counts throughout the whole corpus. Whereas, an unreliable context word will produce different target word co-occurrences depending on the section of the corpus being analysed (Lowe, 2000). To generate a set of reliable contexts words (Lowe and McDonald, 2000) split up the corpus into four equal sections and calculate a co-occurrence matrix for each section. This gives four different column vectors for each context word. They then conduct an ANOVA test (Analysis of Variance) between the context columns for each word and reject words where there is a significant variance between the context columns.

4.3 Latent Semantic Analysis

LSA takes a more globalistic view of word context. Instead of a word being defined by its co-occurrence with other words, words are instead represented by the documents that they are used in (Landauer and Dumais, 1997; Landauer et al., 1998). Therefore, the co-occurrence matrix is a word x document matrix. Documents in theory can be a combination

of multiple topics, so instead of using raw frequency counts, LSA weights each word with its importance in contributing to the content of the document (Landauer et al., 1998). The formula for calculating the strength of the association (importance) of a word w with a document d is given by Dumais (1991):

$$A(w,d) = L(w,d) * G(w) \tag{4.7}$$

Thus the association of a word with a document $A(w,d)$, is defined by its local weight within the document $L(w,d)$, multiplied by its global weight across all documents within the corpus $G(w)$. Nakov, Popova and Mateev (2001) performed analysis on various weighting functions and found the following to be the most successful:

$$L(w,d) = log(f(w,d) + 1) \tag{4.8}$$

$$G(w) = 1 + \sum_{d} \frac{p(w,d) * log(p(w,d))}{log(ndocs)} \tag{4.9}$$

where:

$$p(w,d) = \frac{f(w,d)}{f(w)} \tag{4.10}$$

In these equations $f(w,d)$ is the number of times the word occurred within the document, $f(w)$ is the number of times the word occurred across the whole corpus and *ndocs* is the number of documents in the collection. The advantage of applying the logarithmic function in local weighting function is that it suppresses extremely high frequencies (Nakov et al., 2001). The global weighting function noted here is also referred to as the *entropy* of the word.

The relationships observed by LSA are not just direct associations of word co-occurrence, but are also indirect or *latent* associations that produce similar results to induction. For example, consider two words X and Y that co-occur across a large number of documents. If Y appears in document Z, then LSA will infer that X is also associated with Z even though it never actually appeared within the document Z. It performs this using a technique called Singular Value Decomposition (SVD) and reducing the number of dimensions in the vector space. The psychological motivation behind the emphasis on induction is based on the cognition of school children. Landauer and Dumais (1997) found that around about 3/4 of the comprehension gained when reading a sentence is done by using the knowledge from another sentence. Landauer et al. (1998) explain this using the following example:

Mary is Bobs mother. John is Bob's father and Mary is Ann's Mother.

Once we have read both sentences, we can infer that Ann and Bob are related because they share the same mother even though this is not explicitly expressed in either sentence.

4.3.1 Singular Value Decomposition in LSA

Singular value decomposition performs the following linear transformation on a matrix M:

$$M = U\Sigma V^T \tag{4.11}$$

In this decomposition U and V represent the rows and columns of the vector space as vectors derived from orthogonal factor values. Σ is a diagonal matrix of scaling values that is used to reconstruct the original matrix. The Least Squares Fit theorem states that any matrix can be decomposed perfectly in this way using no more factors than the smallest dimension (Golub and Van Loan, 1996). Using this information, LSA then attempts to reconstruct M using the optimal (fewest) k number of dimensions. It does this by setting all but the highest values in Σ to zero, leaving a diagonal containing the k highest dimensions, Σ^k. M is then reconstructed as $\hat{M}$:

$$\hat{M} = U\Sigma^k V^T \tag{4.12}$$

Landauer and Dumais (1997) explain how this reduced dimensionality infers associations between words. They say that when SVD reduces a word vector it not only uses the information about occurrences of that word across all contexts, but uses a linear combination of the data for every word in the matrix. Therefore, any change to any word counts within the matrix will affect all words. Since we are looking to infer associations, the dimensionality is not based on perfectly reconstructing M, but on producing the correct induction of relations between the words. To find this optimal dimensionality an external test of validity is normally done depending on how the semantic space will be used.

The main disadvantage with SVD is that it is computationally expensive. The Linear Algebra Package (LAPACK) a software library for linear algebra computes SVD to O(mn^2) for a matrix of size m x n where $m \geq n$ (Anderson, Bai, Dongarra, Greenbaum, McKenney, Du Croz, Hammerling, Demmel, Bischof and Sorensen, 1990). Because of this, it may be impossible to compute SVD for some matrices with extremely large dimensions in the computation time given.

4.3.2 Similarity in LSA

Landauer and Dumais (1997) found that Cosine produced the best results for assessing the associativity of two words. Unlike the semantic space produced by Lowe and McDonald (2000), SVD can produce negative values, so the range of the Cosine lies between -1 and 1, with -1 being dissimilar and 1 being similar.

4.4 Cognitive Priming in Semantic Spaces

All three of the semantic spaces discussed were able to find that words produced in free association are more similar than unrelated words (Burgess, 1998; Lowe and McDonald,

2000; Jones et al., 2006) . Therefore they are all able to observe direct cognitive priming. However, out of the three models, only LMS has been able to observe graded and mediated priming by replicating experiments performed by (McKoon et al., 1992) and (Balota and Lorch, 1986). Section 3.3 stated that for our semantic space to have the ability to replicate automatic associations between concepts and attributes, it must be able to observe all types of cognitive priming. Therefore, we could select LMS without any empirical assessment of our own because it is the only model that has published evidence of all three priming effects. However, there are two reasons why we have decided not to do this. The first is that LMS was created using the British National Corpus, which is significantly different to the Enron Corpus that we have selected. Therefore we cannot be sure that the same results will be achieved using the same model. The second is that each type of semantic space is created using unique parameters and no record of studies exist that have attempted to combine features from each. An optimal implementation could be one that combines variant context words used in HAL, with the log odds ratio used by LMS and SVD used in LSA. For this reason, we have incorporated features from each into our design and created an association test that allows us to select the optimal combination of parameters.

4.5 Chapter Discussion and Project Outline

In this chapter we have introduced three approaches one could take in creating a semantic space model of word use. Evidence has suggested that it is possible to observe cognitive priming in such a model. Having introduced the literature surrounding our topic area, we are able to outline the structure of the project:

1. In the first stage of the project we **prepare and pre-process** the Enron corpus, so that it is in a state suitable for natural language processing, by removing unwanted content and separating individual word tokens.

2. Next, we use this pre-processed corpus and **create a semantic space** of word use that incorporates features from HAL, LMS and LSA.

3. The **optimal combination of features** is next selected using a free association test, and validated by proving that it can **observe graded and mediated priming** by replicating McKoon et al. (1992)'s and Balota and Lorch (1986)'s experiments.

4. An *application is then created* which uses this semantic space to compare the association between two groups of concepts and two groups of attributes.

5. Finally, this application is used to **explore cultural attitudes** upheld by the Enron Corporation.

Chapter 5

Enron Corpus

The Enron Corporation, an energy and commodities company, grew to become the 7th largest company in America (by market capitalisation[1]) after only 15 years since being founded in 1985 (Currall and Epstein, 2003; Sims and Brinkmann, 2003). They were regarded extremely highly in the business sector, with Fortune magazine rating Enron as the 'most innovative company' 6 years running between 1996 and 2001 (Fox, 2002; Fusaro and Miller, 2002). The excitement caused by the growth in popularity and success attracted the best and brightest business traders and graduates from around the world. By 2001, Enron employed over 20,000 employees in over 40 countries (Fox, 2002). After a series of investigations into the accounting practices used within the company, Enron was forced to revise their financial reports for the years between 1996 and 2001, revealing a substantial increase in debt and decrease in profits. Shortly after, in December 2001, Enron filed for bankruptcy and were forced to lay off 4,000 employees (Fox, 2002). This sent a massive shockwave through the business community and as public outrage ensued so did investigations into the accounting scandal.

As a result, the Federal Energy Regulatory Commission (FERC) conducted an inquiry and obtained over 600,000 emails from 158 employees working at Enron. The version of the Enron corpus chosen for this project was created by Andrew and Heer (2004) and is available as an SQL dump from *http://bailando.sims.berkeley.edu/enron_email.html*. The Enron Corpus is interesting in the sense that it holds the largest set of emails from a single domain which is publicly available. In its entirety, it includes a sample of word use within a single society of people working at Enron. Therefore performing semantic analysis on the Enron corpus allows us to observe attitudes upheld by the company itself. Before we can commence this semantic analysis, the version of the Enron corpus is described as well as the preprocessing steps we have taken in preparation for semantic analysis.

[1] *Market capitalisation* is the current price of shares times the number of shares outstanding

5.1 The Andrew and Heer (2004) Enron Dataset

Andrew and Heer (2004) originally created the Enron corpus database in order to create a social network of email communication and to categorise emails by content. For this reason a number of tables have been discarded which contain node and edge information from the social network, as well as categories generated from their LSA algorithm. From this database the following tables have been chosen as seen in 5.1 to create an SQL database appropriate for semantic analysis.

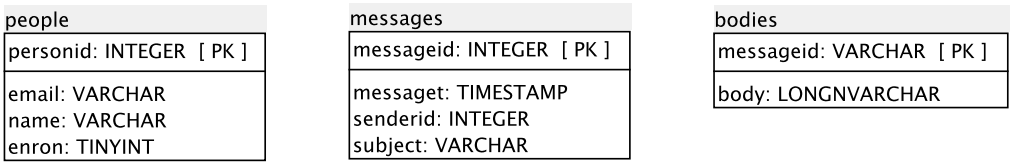

Figure 5.1: Enron database schema

5.1.1 Description of database

1. *bodies*: This contains the entire body of emails separated from headers, timestamps and subject titles.
   - messageid (primary key):The id of the message containing the body.
   - body: The body of the email.
2. *messages* : This contains email information such as the date and time the email was sent, the sender of the email and the subject header of the email.
   - messageid (primary key): The id of the email.
   - messaget: The timestamp of the email.
   - subject: The subject header of the email.
   - sunderid: The id of the person that sent the email, which can be cross referenced with the *people* table.
3. *people*: This contains all of the information from the people extracted from the "Sender:","To" and "CC:" fields of the emails and contains their email address, full name and whether they used an Enron email address.
   - personid (primary key): The id of the individual, which can be used to identify the sender of an email in the *messages table*

- email: The email address of the individual.
- name: The full name of the individual.
- enron: A boolean value that is set to 1 for emails of the form *user@enron.com* and 0 for external email addresses.

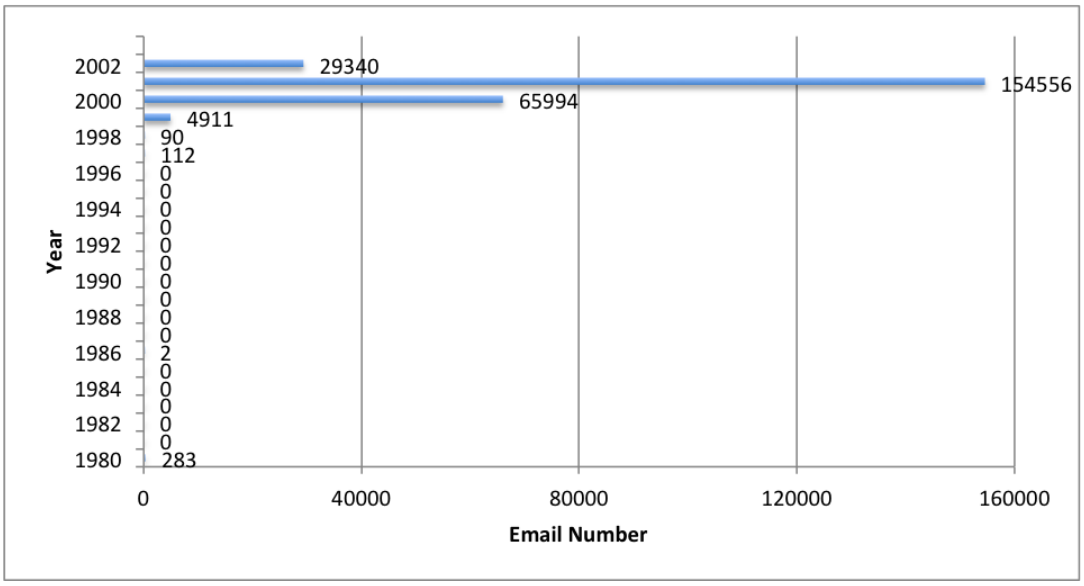

Figure 5.2: Graph of the number of emails sent by year in Andrew and Heer (2004)'s version of the Enron Corpus

5.2 Pre-processing the corpus

Although the dataset used by Andrew and Heer (2004) has already separated the headers from the body of the emails, there is still a significant amount of work that needs to be done before the the email bodies can be used to create a semantic space. In order to prepare the Enron corpus for semantic analysis we have created a Python script (see Appendix B.1), using the PySQLdb library to connect to the database, the Natural Language Processing Toolkit (NLTK) (Perkins, 2010) and other built-in libraries in order refine the raw content of the email bodies. At each stage of the processing, the current version of the corpus was saved and changes were written to a new version of the corpus.

5.2.1 Removing emails

As we are observing the attitudes of Enron as a company, any emails that were sent by individuals that did not work for Enron were removed. There remains no documentation containing information on the employees that have worked for Enron, so we have defined an Enron employee as one that uses the generic "@enron.com" ending in their email address and have therefore removed any emails that were not sent with this type of email address.

As we can see in Figure 5.2, the majority of the emails in the corpus were sent between 1999-2002. The frequency of emails sent outside this time period is questionably sparse, so we have removed all of these emails and kept only those that were sent between 1999-2002.

Finally, all 4495 emails sent by `pete.davis@enron.com` were non-human generated and were also removed from the database. An example of an email sent by `pete.davis@enron.com` is shown below :

```
"Start Date: 2/6/02; HourAhead hour: 21;  No ancillary schedules awarded.
No variances detected.
LOG MESSAGES:
PARSING FILE -->>
O:Portland\WestDesk\CaliforniaScheduling\ISOFinalSchedules\2002020621.txt
!!!General SQL error.
Couldn't update; currently locked by user 'Admin' on machine 'NAHOU-TRDTS5'"
```

5.2.2 Refining the content of emails

A large proportion of email communication occurs in response to another email, and often these responses include the original email. Klimt and Yang (2004) discovered (in his version of the corpus) 63.1% of the emails in the Enron corpus were threads, therefore it is quite likely that there is a large proportion of repeated information in the corpus which could bias our results. In order to ensure this didn't happen we used the regular expression library in Python, *re*, in order to locate any emails that included the following regular expressions and thus remove forwarded information from the emails:

- `.*\s*[Ff]orwarded\sby(.*\n)*`
- `.*[Oo]riginal\s[Mm]essage(.*\n*)*`

Furthermore, we also removed any urls, email addresses, computer generated times (of the form 01:00 or 01:00:00) and dates from the emails using following regular expressions:

- emails: `\S+@\S+\.\S+`
- urls: `(https?://|www.|ftp.)+\S+*`

- timestamp :`^\d\d:\d\d(:\d\d)?$`
- dates: `^\d\d?/\d\d?/\d\d\d\d$`

Finally, we also removed occurrences of *=20* as it occurred frequently throughout the emails and seemed to be a special return character. Furthermore, the combination $=\backslash r\backslash n$ was found in the middle of words that continued onto the next line. In order to join both parts of the word together this combination was also removed.

5.2.3 Tokenising the Corpus

In order to create a semantic space of word usage, one has to be able to extract word tokens from the stream of text found in the email bodies. The following needs to be accomplished to do this:

1. Separate the email body into individual sentences.
2. Separate those sentences into individual words.

In the English language we separate sentences using the punctuation '?', '!' and '.' . However, sentence segmentation is not as simple as separating sentences by using these punctuation marks as delimiters. Kiss and Strunk (2006) illustrates the issue with the following example:

```
CELLULAR COMMUNICATIONS INC. sold 1,550,000 common shares at $21.75 each
yesterday, according to lead underwriter L.F. Rothschild & Co.
```

In this sentence the full stop is used as a decimal place (`$12.25`) and as an abbreviation (`L.F.` and `INC.`). Kiss and Strunk (2006) mention that because of this reason, there can be no formal definition for a sentence which we use can to segment text. Instead, a system called *Puntk* (german for period) has been created, which is an unsupervised classification system. It first traverses through the text and classifies periods as either instances of ellipses, abbreviations or ordinal numbers and assumes that any other occurrences of periods are sentence delimiters. The periods are annotated accordingly and the sentences are separated where there is a period annotated as being at the end of a sentence. Kiss and Strunk (2006) evaluated their system on the Wall Street Journal corpus and found that individual sentences were found with 98.74% accuracy.

Once the text has been split up into separate sentences, each sentence can be divided into individual word tokens. This task is a lot more trivial than the sentence segmentation and often depends on how define separate word tokens are defined. The tokeniser for the

Penn Treebank annotated corpus of English (Marcus, Marcinkiewicz and Santorini, 1993) uses the following conventions:

1. All words are separated by whitespace.
2. Periods are only separated from the word at the end of the stream of text.
3. Contractions are separated into components, e.g *can't* is split into *ca* and *n't*, *could've* is split into *could* and *'ve* and *I'm* is split into to *I* and *'m*.

From this it is plain to see why sentence segmentation is so essential; because we can distinguish whether a period is being used as an acronym or as a sentence delimiter by whether it appears at the end of the sentence or not. In Section 6.1.1 and Section 6.1.2 we mention that our system does not analyse contractions and therefore words such as *can't* and *won't* are ignored and have no impact on the semantic space.

In order to tokenise the corpus we have used the NLTK library for Python as it provides an implementation of both the Punkt sentence tokeniser and the Penn Treebank word tokeniser. Whilst tokenising, single quotation marks have been removed that appear in any word (except for n't, as we have used this token to find negative uses of words in Section 5.2.5). The was done because the tokeniser can provide some unwanted behaviour if a word is put inbetween single quotation marks, e.g *'cool'* will be separated into *'cool* and *'*. Any tokens that only include punctuation have also been deleted; we hypothesise that punctuation gives little semantic context to a word and by removing punctuation we condense the corpus and reduce the computation time needed to traverse the corpus.

After tokenisation, each token was written back into the corpus separated by whitespace. In doing this, when the corpus needs be traversed to collect co-occurrences, we can separate the tokens easily using the whitespace as a delimiter rather than performing tokenisation every time; thus reducing the computation time.

5.2.4 Lemmatising the corpus

In natural language a single word can be used in many inflected forms. For example, the verb *to talk* can be represented as *talk*, *talked*, *talking* and *talks*, the adjective *quick* can be represented as *quick*, *quickly*, *quicker* and *quickest*. All of which represent the same words *talk* and *quick* and have the same semantic meaning. Dumais (1991), Lund et al. (1995), Landauer and Dumais (1997) and Lowe and McDonald (2000) all agree that in order to a create reliable representation for a word, you need as many occurrences of that word as possible. By reducing *talk, talked, talking* and *talks* to the same root word or lemma *talk* we increase the number of occurrences of the root word and therefore the reliability of the representation provided by the semantic space. This process of reducing an inflected form to a root word is called *lemmatisation*. In order to do this we need knowledge of all root words in the English language and their different word senses. WordNet (Miller, Beckwith,

Fellbaum, Gross and Miller, 1990; Miller, 1995) provides us with this information.

WordNet is a lexical database of the English language and contains over 110,000 different word forms (Miller, 1995) and their links to their synonyms (same-meaning), antonyms (opposite meaning), hyponyms (subordinate), hypernyms (superordinates) and root lemmas. Within the database separate representations exist for the each sense that a word can have and they are categorised by whether they are a noun, verb, adjective or adverb (Miller et al., 1990). WordNet can be accessed through the NTLK library and contains an implementation of lemmatisation. However, the prerequisite for its lemmatisation is that each word needs to be annotated with its lexical classification; whether the word is a noun, verb, adjective or adverb.

Lexical classification annotation is referred to as *part-of-speech* tagging (or POS tagging). The most popular method for this is to train a Hidden Markov Model (HMM) with a corpus that has been pre-tagged with the lexical classification for each word (Hasan, UzZaman and Khan, 2007). Given the HMM, the POS tagger then finds the tag that is most likely to occur with the word it is presented with. The most simple type of HMM is a Uni-Gram, which only considers the word itself and not the surrounding context. A tag will be chosen because it is the most frequent tag that occurs with the word in the pre-tagged corpus, for example the tagger will classify a word as being a verb because it appears more often as a verb than it does as a noun. An N-Gram however, considers a sequence of n tags that precede the word and tries to find a tag sequence that maximises the following formula given by (Hasan et al., 2007):

$$p(word|tag) * p(tag|previous\, n\, tags) \tag{5.1}$$

That is, the probability of word given the tag, multiplied by the probability of the tag, given the proceeding n tags in the sequence. Examples of N-Grams are Bi-Grams, which only consider one tag which precedes the word, Tri-Grams, which consider two tags preceding the word, and Quad-Grams, which considers three. Although the correctness of a tagger increases as it considers more tags that come before a word, a problem can occur when it reaches a sequence of tags that it doesn't recognise, at which point it can't annotate the word. A simple solution is to use what is called a *back-off tagger* (Perkins, 2010). This is where the tagger contains multiple N-Grams in a hierarchy, it first attempts to tag the word based on N-Gram at the top of the hierarchy and if it can't tag the word with the current N-Gram, it moves down the hierarchy until it finds a N-Gram that can. The NLTK library contains classes that allow a simple implementation of a back-off tagger as shown below:

```
def backoff_tagger(trainingdata, tagger_classes, backoff=None):
        for cls in tagger_classes:
                backoff = cls(train_sents, backoff=backoff)
        return backoff

tagger = backoff_tagger(trainingdata, [UnigramTagger, BigramTagger,
```

```
TrigramTagger], DefaultTagger('NN'))
```

In the code above, *backoff_tagger* first creates a *UnigramTagger* and sets its back-off tagger to a default tagger which tags all words as nouns. It then creates a *BigramTagger* and sets its back-off to the previously created *UnigramTagger*. Finally, it creates a *TrigramTagger* with the *BigramTagger* as its back-off tagger. When this tagger reads a word it will first use the *TrigramTagger* to tag the word, if this is unsuccessful it will resort to its back-off tagger - the *BigramTagger*. If the *BigramTagger* is also unsuccessful it will resort to its back-off tagger which is the *Uni-gram tagger* and finally, if this is unsuccessful it will just tag the word as being a noun.

We have used this implementation in creating a POS-tagger for the corpus and have trained it on the pre-tagged Brown Corpus which is also available in the NLTK distribution (source code can be found in Appendix B.2). The Brown corpus was originally created in 1961 and contains roughly 500 samples of English text and just over a million words that have been POS-tagged by hand (Francis and Kucera, 1979). Once trained, we have then applied this tagger to the whole corpus, feeding each tagged word into the WordNet lemmatiser and replacing it with its root lemma.

5.2.5 Final Stage of Preprocessing

In the final stage of the preprocessing, we have traversed through the whole corpus and removed any emails containing less than 10 word tokens. We hypothesise that a large amount of emails will be short replies such as "Thanks for letting me know". These emails do not contain useful information about word use and could have a negative effect on the final results of the semantic space.

As well as this we attempted to remove any *negated words*. In language the word *not* or contractions ending in *n't* are used to convert a word into its opposite meaning. This could pose a threat to the reliability of our semantic analysis. For example consider the word *happy*, as well as positive uses of the word, our system would also record negative uses. Therefore the system will incorporate meanings such as *isn't happy* or *not happy* into the word. However, these do not mean the word *happy* but the word *sad*. As a precaution, the negative tokens *not* and *n't* were therefore identified (remembering that the Treebank tokeniser splits contractions like *can't* into *ca* and *n't*, see Section 5.2.3). We then found the first *noun*, *verb*, *adjective* or *adverb* that succeeded the negative token and tagged the word with the prefix *not-*. For example the preprocessing finds sentences including *not happy* or *isn't happy*, and replaces happy with 'not-happy'. When performing semantic analysis in Chapter 6 any negatively tagged words are ignored and therefore do not distort the meaning of the word.

5.3 Analysis of Pre-processed Corpus

The final version of the Enron corpus contains 148,323 emails, a total of 22,184,645 words, and 222,543 unique word tokens. To assess the performance of the pre-processing and tokenisation we created a script (see Appendix B.3) to count the proportion of *useful* tokens in our corpus. We define useful words as those that occur in the English, American or WordNet dictionaries. From the rest of the words we also found the proportion of those that were male or female names (by using the Names corpus in NTLK), tokens not recognised by the dictionaries and non-words (combination of characters, numbers and punctuation).

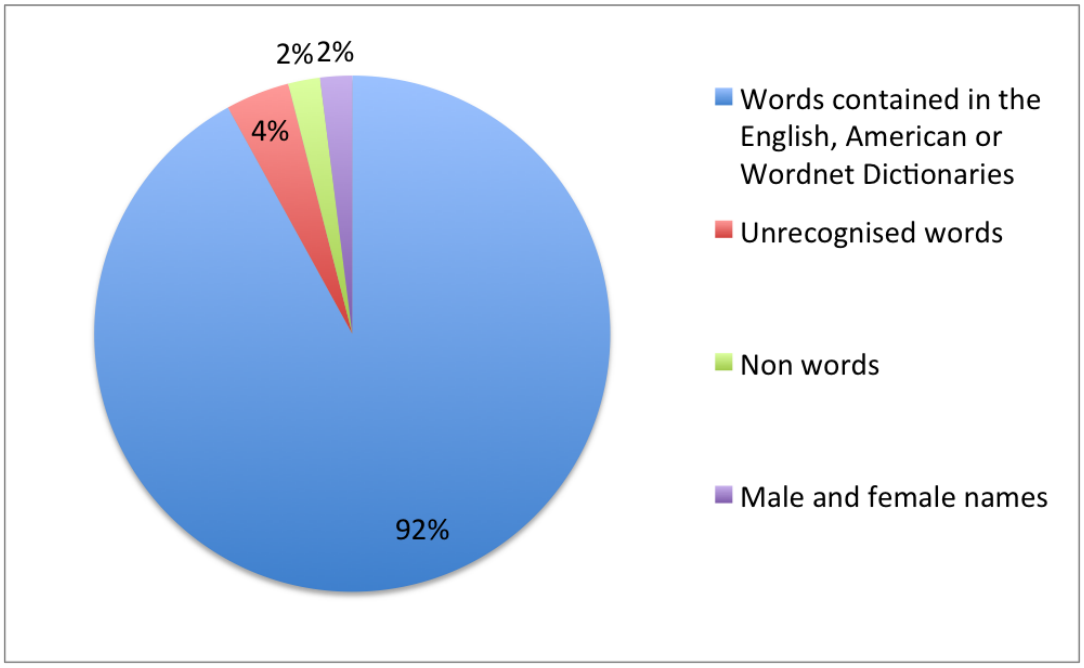

Figure 5.3: Piechart of the proportion of different types of tokens in the pre-processed Enron Corpus

.

Figure 5.3 shows that 92% of the tokens in the corpus are useful, 4% of the tokens in the corpus are words unrecognised by dictionaries, 2% are male and female names and 2% are non words. This high percentage of useful tokens allows us to conclude that the pre-processing of the corpus is a success. Furthermore, we can also verify that the Enron corpus contains enough rich information that can be used in natural language processing such as creating a semantic space. The figure also identifies that further work could be done to increase the percentage of useful tokens. For example, although we expect names to be used frequently within emails, these normally occur at the start and end and in email

signatures. Therefore by removing uses of names that occurred at these points, a large amount of redundant information and content that is not relevant to understanding word use would be removed. Furthermore, the higher than expected percentage of non words could suggest that there is still a significant amount of non human generated emails which need to be identified and removed.

5.4 Chapter Disscussion

In this chapter we have discussed the many steps taken in pre-processing the Enron corpus. In doing so we have created an SQL database of tokenised emails that contain a significant proportion of information that is useful for natural language processing. The next chapter discusses the Enron Semantic Space which has been created in order observe cultural attitudes.

Chapter 6

The Enron Semantic Space

Using the pre-processed corpus mentioned in the previous chapter we have created a model of the way in which words are used in Enron emails; this we call the *Enron Semantic Space*. We remind the reader that a semantic space is a *target word* x *context* matrix of co-occurrences. The Enron Semantic Space is influenced by all three models discussed in Chapter 4: HAL (Lund et al., 1995), LMS and LSA (Landauer and Dumais, 1997). As the Enron corpus has never been represented as a semantic space before we kept the scope of the initial design fairly broad. Rather than focusing specifically on one implementation, we incorporated many features of each of the models in the design as we could not assume that one would perform better than another.

In the first part of this chapter the initial design is described: the type of *target words* selected for use, how the *context* of word use is designed, the algorithm used to capture *target word-context word* co-occurrences and finally the various types of parameters incorporated into our model.

The second part focuses on refining our implementation, by selecting the parameters that perform the best on a free association test (or direct cognitive priming test). We then prove that this refined model can observe both graded and mediated priming and is therefore an appropriate model for use in discovering attitudes.

The implementation of the Enron Semantic Space was designed in Python 2.6, using pymysql, numpy, scipy and NTLK and can be found in Appendix B.4. The large number of libraries available for Python has made it an appropriate choice of programming language and has eased the implementation process. The combination of both object-oriented and procedural paradigms used in Python has also given us a great amount of flexibility in the way which the can be the model can be designed.

6.1 Initial Design and Implementation

There have been a variety of studies which have analysed different parameters for designing a semantic space. In these studies the *Academic American Encyclopedia Corpus* (Landauer and Dumais, 1997), the *British National Corpus* (BNC) (Levy, Bullinaria and Patel, 1998; Lowe, 2000; Bullinaria and Levy, 2007), and the *USENET Corpus* (Lund et al., 1995; Lund and Burgess, 1996) are some of the corpora that have been studied. One might presume that because of this there is little room for exploration when designing a semantic space for the Enron corpus. However this is not the case as none of the studies have modelled the Enron corpus. Also the corpora that have been modelled are significantly different in size and content:

- **Enron Corpus: 22,184,645 words from 148,323 emails.**
- British National Corpus: Over 100 million words taken from a selection of written (books, periodicals, unpublished) and spoken text (lectures, interviews, legal proceedings) (Aston, 1997).
- USENET Corpus: 160 million words taken from the USENET groups internet forum (Lund et al., 1995; Lund and Burgess, 1996)
- Academic American Encyclopaedia Corpus: 4.6 million words from Grolier's Encyclopaedia for young students (Landauer and Dumais, 1997)

Bullinaria and Levy (2007) demonstrated that the behaviour of semantic spaces varies with different size corpora. As the Enron corpus is significantly different in size to the other corpora we cannot make assumptions about parameter performance from their studies. The Enron corpus significantly differs also in the variety of content that it contains. The BNC, USENET and Academic Encyclopaedia all cover a wide variety of topics (social, political, educational etc.) and therefore a variety of word use. On the other hand, the Enron corpus contains emails from a very specific domain and one can be fairly confident that the majority of emails will be related to business. Consequently, there may be a significant bias in the way words are used. Therefore even if the corpus was the same size one can't be certain that it would behave in the same way. Finally, the information contained in both the BNC and the American Encyclopaedia is clean and edited and probably far more reliable than the information in the Enron corpus. Even though a lot of processing has been carried out, there are still a significant amount of non-words. Whereas, there will be few of these in the BNC and American Encyclopaedia as all of their documents will be human generated.

To ensure our model is the most appropriate, we therefore have to remain naïve about the optimal features to use. First we ensure that our semantic space is flexible and can implement a few different types of each feature; only after we have gathered empirical evidence can the optimal features for the model be selected. Due to the scope of the project it is unrealistic to consider all combinations of features so there has to be a few features which are fixed. Before commencing the description of different types of features that have been

analysed, those which we have fixed will first be described, starting with the type of target words, then the definition of context and finally the method for capturing co-occurrences.

6.1.1 Choice of Target words

When analysing textual meaning it is imperative to isolate the types of words to be studied. In linguistics, words are divided into two categories: *content* words and *function* words. All content words contain a detailed amount of semantic information. Used within a sentence they are able to name objects (nouns) or events (verbs) and describe those objects and events (adjectives and adverbs). For this reason they are often seen as being at the heart of a sentence's meaning (Corver and Van Riemsdijk, 2001). Function words on the other hand are less conceptual, they are the set of words that perform the function of being the glue that holds a sentence together. Function words are split into prepositions (in, on, at, between), determiners (the, at, my, more), conjunctions (and, that, when, while), modal verbs (can, must, will, should), auxiliary verbs (is, have, got, do), articles (no, not, as) and pronouns (he, she, they, we, you) (Pennebaker and King, 1999).

Out of the groups of function words, only pronouns can be considered to contain useful semantic meaning; for example *he*, *she*, *his* and *her* refer to gender and *me*, *we*, *you* and *they* refer to the view of self or others. Consequently, all content words are allowed to be target words, but pronouns are included from the set of function words.

Target words must also be chosen so that they occur frequently enough in the corpus to provide reliable results (Landauer et al., 1998; Lowe, 2000; Lowe and McDonald, 2000). Words that occur infrequently over the corpus will not only provide little information about their use but are also likely to be less resident to extraneous use (for example when a word is accidentally used out of context). As we have have mentioned in section 4.2.1, Zipf's law states that most words occur very infrequently within text; so as we limit the target words to only those that meet a certain frequency threshold we also greatly limit the amount of words which can be analysed.

In order to study attitudes in the Enron corpus, we therefore need to obtain a set of target words that are reliable enough so as not to distort our findings and large enough to provide a sufficient number of words to represent concepts (for example enough words that represent *pleasant* and *unpleasant* meaning). Lowe and McDonald (2000) set this threshold to words that occur at least 100 times in the BNC, however because the Enron corpus is smaller in size we found this far too restrictive and removed a large proportion of the words that we needed to analyse. Instead, the threshold was set to 50 which included an acceptable amount of target words.

To extract the set of content words we used the FreqDist (Frequency Distribution) class within the NLTK library to create a hash-table or dictionary of frequencies of each unique token that appeared in the corpus. We then selected target words that:

- had a frequency >= 50; and
- were defined in either the PyEnchant American, PyEnchant English or Wordnet Dictionaries, the set of pronouns; and
- were not contained in the corpus of function words provided by NLTK (called stop words).

FreqDist was chosen as it automatically sorts itself by frequency; with the most frequently occurring words appearing at the start of the dictionary. This meant we could iterate through the dictionary until a word was reached that occurred less than 50 times. This procedure (found in Appendix B.3) has allowed us to extract a set of 8,199 target words.

6.1.2 Choice of Context

As we have mentioned in Section 4 there are two ways in which we can represent the context of word use:

1. As words that neighbour the target word (HAL, and LMS).
2. As emails in which the target word appears in (LSA).

In the Enron Semantic Space we have chosen the first option: to represent context as words that co-occur with the target word within in a fixed window; thus creating a matrix of *target word-context word* co-occurrences. Unlike our target words, we only consider content words and not function words as defining the context of word use. The reason for this is function words provide a syntactical definition of word use rather than a semantical definition (Padó and Lapata, 2003). This would skew the representation of words towards its grammatical use rather than its semantic use. If one defined a word by its grammatical use then associative strengths between words could not be obseved, as two words can be associated but be used in completely different syntactical contexts. An example of this is the association between *sea* and *swim*; they are both related to each other but are found in different syntactical contexts as *sea* is a noun and *swim* is a verb.

It was decided not to include pronouns in the set of context words because of Zipf's law (Section 4.2.1). Pronouns, like all function words, occur far more frequently than content words (Pennebaker, Mehl and Niederhoffer, 2003) and are equally likely to occur with all words. Hence, we have not included pronouns as we postulate that they will not differentiate significantly in their occurrence with different function words.

The reason it was decided not to represent context as emails is that all emails in the corpus would have to be considered. Given that we have established a set of 8,199 target words and there are 148,323 emails in the corpus this would mean that we would have to create a 8,199 x 148,323 sized matrix. This is too large to create on 4GB worth of RAM

used in this project. In order to feasibly create a matrix of this size, a distributed memory architecture would have to be utilised and semantic analysis in parallel on multiple cores be performed; this is out of the scope of the project.

6.1.3 Capturing Co-occurrences

The algorithm for capturing the co-occurrences of target words and context words will now be briefly discussed. This is carried out using a fixed window technique in a similar fashion to Lund et al. (1995) and Lowe and McDonald (2000) seen in section 4.1 and section 4.2 respectively. Each email is read into the corpus and split into a list of tokens that maintain the order of the words in the email; a window is then passed through the email and analysed. The type of window that we use is a bi-directional window in order to capture co-occurrences of contexts words before and after the target word.

Words at the head and tail of the window are first checked to see if they are in the set of target words to be analysed. If the head of the window is a target word then all other words in the window are checked to see if they are context words. For each context word that it finds, the value for the *target word-context word* pair in the co-occurrence matrix is incremented. This is done similarly for the tail of the window. As the window passes through the email every word becomes the head and tail of the window, therefore words that appear before and after the target word are analysed. The words that are considered to co-occur are those that appear within *window size - 1* words either side of the target word.

For example consider the window of size 5, [*hi*,*how*,*are*,*you*,*today*]:

- *hi* and *today* are both checked to see if they are target words.
- if *hi* is target word then *how*, *are*, *you* and *today* are checked to see if they are context words and co-occurrence counts are updated accordingly.
- if *today* is a target word then *hi*, *how*, *are* and *you* are checked to see if they are context words and co-occurrence counts are updated accordingly.

In Section 6.1.4 we describe two ways that co-occurrences can be updated in our naïve design; using either a flat window or a weighted window. In a flat window all words are weighted equally and therefore are all incremented by a value of 1 for each occurrence. In a weighted window words are given a higher value the closer they are to the target word; this value is: *length of the window - distance from the target*. Algorithm 1 describes the process of extracting co-occurrences from a window of words.

Algorithm 1: Analyse Window

Data: Window of words
Result: Co-occurrence counts are updated for tail and head target words

```
headistarget = true if head is a target word, false if not;
tailistarget= true if tail is a target word, false if not;
if headistarget or tailistarget then
    valuefirst=1;
    valuelast=1;
    foreach i to length(window) do
        if weighted then
            valuefirst=len(window)-i;
            valuelast=i+1;
        end
        if i!=0 and headistarget then
            Update co-occurrence matrix for window[i] and head
        end
        if i!=length(window)-1 and tailistarget then
            Update co-occurrence matrix for window[i] and tail
        end
    end
end
```

In order to store co-occurrences, three datatypes are used: a 2-d array of co-occurrence values, a hash-table linking target words to rows and a hash-table linking context words to columns. When the co-occurrences for a target word and a context word need to be updated, the row hash-table is queried with the target word as the key; this returns the row of the target word. The column hash-table is also queried with the context word as the key; this returns the column of the context word. Once we have the row and the column of the *target word-context word* pair we can increment value of the cell in the 2-d array at those indexes. At first glance it may seem that this method is including redundant information needlessly, i.e why not just have one hash-table, that links target words to a hash table of context words? The reason for this is that the Singular Value Decomposition that we perform in 6.1.4 needs to have the data in an array form.

Taking a step back, we will now describe how we pass a window through all emails in the corpus. Rather than querying the database and storing all of the email bodies client side, we use a server side cursor. This is implemented using the SSCursor class in PyMySQL which stores the queried email bodies in a buffer on the server, where each email can be loaded from the buffer separately into memory. A window is then passed through each email in the corpus using a *queue*.

Once an email is read and tokenised (by using whitespace to separate the tokens) the window is initialised to contain a queue of empty strings: [“”, “”, “”, “”, “”, “”]. The first token is then read and pushed onto the end of the queue and the empty string at the start of the queue is popped: [“”, “”, “”, “”, “hi”] . At this point the window is analysed for any co-occurrences. This process occurs until all tokens are read into the window; at which point the tail of the window has reached the end of the email: [“hi”, “how”, “are”, “you”, “today”]. When the window has reached this point an empty string is pushed onto the queue, the item at the front of the queue is popped and the window is analysed again: [“how”, “are”, “you”, “today”, “”]. Empty strings are then pushed onto the queue until the head of the window reaches the end of the email [“today”,“”,“”,“”,“”] and the final analysis on the email is performed. Algorithm 2 represents this process of passing a window through each email in the server side email buffer. Once all emails have been read we have a fully populated co-occurrence matrix.

Algorithm 2: Capture Co-occurrences

Data: Email buffer

Result: Co-occurrence matrix is populated with co-occurrences for all emails in enron_corpus

```
foreach email in emailbuffer do
    tokenlist = tokenize(email);
    window = initilise queue of empty strings;
    foreach token in tokenlist do
        Pop word from front of window;
        Push token onto back of window;
        Perform windowAnalysis on window;
    end
    while window[0]! = emptystring do
        Pop word from front of window;
        Push word onto back of window;
        Perform windowAnalysis on window;
    end
end
```

6.1.4 Model Parameters

Now that the features that are fixed in our semantic space have been described the parameters that are varied in our naïve implementation can be introduced. Halmos (1947) describes a semantic space as a tuple $< B, A, M, S >$, where:

- B = set of basis elements $(b_1, b_2, ...b_n)$ where n defines the dimensionality of the space.

- A = a function that represents the association between a target word t and a basis b_i so that a target word can be represented as a vector $v = [A(b_1, t), A(b_2, t)..., A(b_n, t)]$.
- M = a mapping from the semantic space to space of lower dimensionality.
- S = a similarity metric for comparing the similarity of two target word vectors.

All three models described in Chapter 4 derive a different interpretation as to what each element of this tuple should represent. Thereby there are different ways each element can implemented. The parameters for each item in the tuple will now be identified and incorporate into our naïve implementation.

A: Association function

Influenced by: HAL and Lowe and McDonald (2000)'s Space
Parameters:

- Window type : weighted or flat.
- Window size
- Lexical function: normalisation or log odds ratio.

The *association function* is split into three parts. The first two parts the are window type and window size that are used to capture raw frequencies of *target word-context word* co-occurrences. The final part is the lexical function that takes raw frequencies found in the window and produces the final value of association between the two words.

As we mentioned in Chapter 4, HAL and LMS use different types of windows. In LMS the window used is a flat window where all context words in the window are given the same frequency value. Conversely, HAL uses a weighted window where context words that are nearer to the target word are given a higher frequency value. Bullinaria and Levy (2007) suggest that weighted windows might contain more syntactic information about a word than semantic and thus might be less appropriate. In a sentence nouns that are adjacent to each other are rarely found. If we ignore function words, they are normally separated by at least one word of another lexical type (verb, adjective, adverb). When considering a word vector for a noun captured using a weighted window, it will have higher co-occurrence values for context words that are verbs, adjectives and adverbs as they will appear closer to the word than other nouns will. This means that words of the same lexical category will be found to be more similar; two nouns will be similar in the way they weight adjectives, verbs and adverbs highly compared to nouns.

Levy et al. (1998) also suggests that the type of window greatly affects the size of the window that should be used. If a flat window is chosen then the window needs to be of a

smaller size to prevent the likelihood of extraneous or unrelated context words appearing within the window. However if a weighted window is used, then one can be more liberal with the window size because even if unrelated context words appear within the window, they will appear further away from the target word. This means they will be weighted lowly and cause less disruption to the data. So as well as finding the most appropriate window type, the optimum window size that corresponds with that window type must be found.

Finally after we have gained frequency counts, there needs to be some lexical function that turns these raw frequencies into association values. We are reminded that HAL and LMS use two different lexical functions and we will include both functions in our naïve implementation. In HAL a simpler function is used called normalisation, where the frequency value for each *target word-context word* pair is divided by the sum of the co-occurrences of the target word with all context words. Afterwards, the sum of the target word vector is equal to 1. Lowe (2000) proved that just using normalisation can still lead to a frequency bias, where words that appear with the same frequency in the corpus are rated more similarly. The lexical function that they use is the log odds ratio (see Section 4.2.1), which has shown to reduce the affect of this frequency bias.

B: Basis

Influenced by: HAL and LMS
Parameters:

- Context word sample method: most frequent, most variable or most reliable.
- Context word sample size.

In Section 6.1.2 content words were identified as the context used in our model, but the sampling of context words to define our basis was left unmentioned. There are three options in sampling context words from our space: one can select the most frequent, most variable or most reliable content words.

The main reason for using the most frequent content words in our space is that we can be sure that all of the co-occurrence counts in the space will be statistically reliable (Lowe, 2000). Although this is the case, more frequent words could co-occur with the majority of the target words in our model. This could result in all target words having extremely similar vectors, meaning we would be unable to distinguish any semantic difference between the words. HAL tries to get around this by first performing semantic analysis on the corpus with all the possible context words, then only selecting those that have the most variant behaviour across all target words. This solves the problem of having similar vectors but also introduces a new problem: words could be introduced that occur too infrequently or are used inconsistently across the corpus, resulting in differences between words that are unreliable. So instead of using the most variable word vectors, LMS selects the words

that behave reliably and consistently across all sections of the corpus. Although choosing the most reliable words seems the most intuitive approach, there exists no empirical evidence that has compared it to using the most frequent or variable content words. All sampling techniques were therefore implemented and the method that produced the best performance regardless of intuition was chosen.

As well considering sampling methods, one must deliberate between the number of context words and thus the number of dimensions to use in the space. This a trade-off between bias and variance. When few dimensions are used, we force a word to be defined by a small sample of its contexts and therefore we bias our model towards those contexts. As more dimensions are introduced the words are represented across a variety of contexts, which allow our model to be more flexible. However this comes at a price, by introducing too many dimensions we again include infrequently used words which could harm the reliability of our data.

M: Mapping

Influenced by: LSA
Parameters:

- Number of dimensions (k) to reduce the space to.

In LSA, SVD is used in order to observe indirect associations between words when using documents as context in a corpus. However, we have yet to find a study that uses SVD when words are used as context in the corpus. Landauer and Dumais (1997) found that by mapping the semantic space onto a low dimensional space, it improved its performance on a synonym test called the Test of English as a Foreign Language (TOEFL). Levy et al. (1998), mention that by using large enough corpora (larger even than the BNC) dimensionality reduction is unnecessary. As the Enron Corpus is far smaller than the BNC it is hypothesised that SVD could significantly improve our semantic space. Landauer and Dumais (1997) also describe how care must be taken when reducing the dimensionality of data. If the dimensionality is reduced too much, then a significant amount of information will be lost and our representation will be overfit. There is no exact method for finding the perfect number of dimensions, so one must instead observe the number of dimensions that produce the best results by varying the value of k dimensions that the space is reduced to.

S: Similarity Metric

Influenced by: LSA, HAL and Lowe and McDonald (2000)'s Space
Parameters:

- Similarity metric: Euclidean, Cosine, City Block, Hellinger, Kullback-Liebler divergence

So far we have seen two similarity metrics that can be used to measure the similarity between words. HAL uses the Euclidean distance; LSA and LMS use Cosine. We now introduce three more similarity functions that have also been suggested by Levy et al. (1998) and Bullinaria and Levy (2007):

City Block : $\sqrt{\Sigma(U_i - V_i)^2}$

Hellinger : $\Sigma(\sqrt{U_i} - \sqrt{V_i})^2$

Kullback-Liebler Divergence : $\Sigma U_i \log(\frac{U_i}{V_i})$ (Where U and V are target word vectors)

Levy et al. (1998) and Bullinaria and Levy (2007) come to different conclusions as to which method produces the best results. In Levy et al. (1998)'s paper, it was found that City Block, Hellinger and Kullback-Liebler all produced the best results when used on the TOEFL test and Cosine and Euclidean produced poor results. Bullinaria and Levy (2007) found that conversely Cosine produces the best results which supports LSA and LMS. As these two studies contradict each other, analysis was carried out to find the most appropriate distance measure.

6.1.5 Naïve Implementation

Now that the parameters that we have been left open to empirical analysis have been described, the naïve implementation will be discussed. To compare the similarity between two words the following steps are taken by our program:

1. **Input**: Database connection, set of target words , set of context words (B).
2. Initialise co-occurrence matrix, row and column index hash tables.
3. Query the Enron corpus on the database connection for all email bodies.
4. Capture the association of target words and context words over all the emails (A):
   (a) Pass a window of specified size and type through each email capturing raw co-occurrences.
   (b) Then convert all the raw frequencies into an association representation using the lexical function.
5. Take the association matrix and map it to a matrix of lower dimensionality (M).
6. **Output**: A semantic space that can be used to compare the similarity between two words using a specified distance metric (S).

6.2 Empirical Analysis of Model Parameters

To assess the performance of different parameters in our naïve implementation we remind ourselves of the purpose of our model. That is the ability to observe cognitive priming and therefore be able to extract attitudes upheld by Enron employees. The simplest form of cognitive priming we have discussed is direct cognitive priming, where a directly associated word primes the target word. If our model is able to observe direct cognitive priming then two associated words should produce higher similarity values than two unrelated words. Thereby an *association test* that measures how accurately our system predicts that two words are directly associated has been designed.

6.2.1 The Free Association Test

The association test is based on the distance test used by Bullinaria and Levy (2007). In the test one is provided with 1000 <*target word-associated prime*> pairs, and for each target word 10 words are randomly selected from other pairs to represent unrelated words. Similarity values are then calculated for the <*target word-associated prime*> pair and the 10 <*target word-unrelated word*> pairs for all 1000 target words. From this we work out the percentage of <*target word-unrelated word*> pairs that produce a lower similarity value than the <*target word-associated prime*> pair, then calculate the average of all the similarity percentages of the <*target word-associated prime*> pairs. The resulting percentage portrays the accuracy that our system will find an associated prime to be more similar than an unrelated word. The higher the accuracy of our system, the better it observes direct cognitive priming.

To acquire <*target word-associated prime*> pairs the largest collection of free association norms readily available was utilised: The University of South Florida Free Association Norms (USFFAN) (Nelson, McEvoy and Schreiber, 2004). Free association norms are generated from an automatic word association test, where participants are asked to respond with the first word that comes to mind when presented with a target word. Words that are generated as responses with high frequency across a range of candidates are said to be free association norms. USFFAN contains almost 0.75 million responses to 5,019 target words from more than 6,000 individual participants, and is available to download in XML format from `http://w3.usf.edu/FreeAssociation/`. Each <*target word-association response*> pair is also associated with its probability value; we randomly selected 1000 of these pairs that both occur with high probability in the USFFAN database and occur at least 50 times in our corpus to use in the free association test.

6.2.2 Structure of Empirical Testing

As mentioned in section 6.1.4 the aim was to arrive at a model that finds the combination of *association function*, *basis*, *similarity metric* and *mapping* parameters which give the

highest accuracy on the association test. Each tuple item has at up to three different types of parameters that have to be established; thus we are unable to observe every combination of parameters for all tuple items as this would be an extremely labour intensive process. To resolve this issue a linear flow architecture for our testing was implemented:

$$A\text{: Association Function Test} \rightarrow B\text{: Basis Test} \rightarrow S\text{: Similarity Metric Test} \rightarrow M\text{: Mapping Test}$$

In this architecture optimum results are collected at each stage and are fed into the following stage as fixed variables. In a pilot study we found that the results from each test were mutually exclusive, such that the optimum parameters where not affected when the order of these tests were changed. These tests and their results in turn will now be described.

6.2.3 Association Function Test

The first test performed was to find the optimal parameters for the association function, as it was predicted that this would have the greatest impact on our system. In doing so we established the combination of *window type*, *window size* and *lexical function* that produced the best results on the association test.

Experimental Setup:

A semantic space was created for each combination of the following parameters:

- Window size: 2-30.
- Window type and lexical function pairing: flat window with normalisation, flat window with the log odds ratio and weighted window with normalisation.

Where every semantic space had the following fixed features:

- Basis: all content words with a frequency ≤ 50.
- Similarity Metric: Cosine.
- Mapping: All k dimensions (no dimensionality reduction).

As the weighted window distorts the frequency co-occurrences it cannot be used alongside the log odds ratio, which is why this pairing was not considered. Each semantic space was then ran on the free association test and accuracy percentages were obtained.

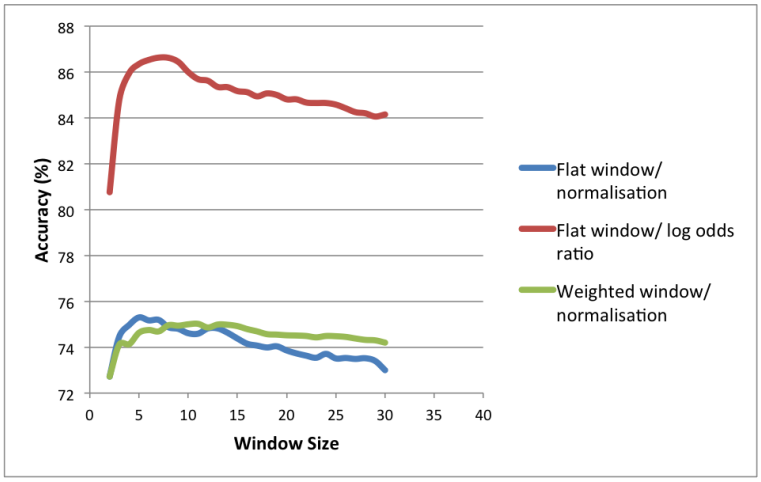

Figure 6.1: Accuracy Results from the Association Function Test. Graph shows the percentage of free associated words that were found for different semantic spaces when the window type, window size and lexical function were varied

Results and Analysis

Figure 6.1 shows the results from the Association Function Test where the optimal parameters were found to be:

- Window of size 8; combined with
- Flat window and log odds ratio; giving
- Accuracy = 86.62%

The graph clearly demonstrates that the most influential parameter is the lexical function as the log odds ratio pairing scored far higher compared to both normalisation pairs: 75.3% for flat window with normalisation and a window of size 5, 75.02% for weighted window with normalisation and a window of size 11. This suggests that frequency bias not dealt with by normalisation is a major factor in distorting the accuracy the semantic space, which is consistent with views held by Lowe (2000). *Pointwise mutual information* is regarded as being equivalent to the log odds ratio and was also found to produce successful results in Bullinaria and Levy (2007).

This graph illustrates the theory stated by Levy et al. (1998), that flat windows require smaller window sizes than weighted windows (also discussed in section 6.1.4) as they are more susceptible to extraneous context co-occurrences. This is shown by the fact that the best window size for the weighted window pairing is 11 compared to a window size of 5 and 8 for the flat window pairings. Moreover, both flat window lines degrade more steeply

than the weighted window implementation when the window sizes increases. The fact that the optimum window size for the *flat window/log odds ratio* pairing is larger than when the flat window is normalised further suggests that the log odds ratio is indeed able to cope with chance co-occurrences that occur in larger windows.

The final interesting point is that when the same lexical function is used there is barely any difference between the accuracy of either flat windows or weighted windows. This suggests the fear that weighted windows includes too much unnecessary syntactical information, as it does not seem to affect the outcome of the results. The only reason for preferring flat windows over weighted windows is that flat windows are more flexible and can be combined with better lexical functions such as the log odds ratio.

We used the association function combining a flat window of size 8 with the log odds ratio in the following experiments.

6.2.4 Basis Test

After the optimum parameters for the association function were established we next identified whether the results from this test could be improved through careful sampling of the basis (context words). As discussed in Section 6.1.4 there are three ways in which we can sample the context words: by choosing the most frequent, the most variant or the most reliable vectors.

To perform the basis test, we first sorted the list of the context words used in Section 6.2.3 by frequency, variance and reliability. To find the variance of each context word a semantic space was created from the Enron corpus using the optimal parameters found in Section 6.2.3. Similarly to Lund et al. (1995) and Lund and Burgess (1996), the variance of the co-occurrences for each context word column was then calculated in turn and the words were sorted by those with highly variant column vectors to lowly variant column vectors.

To calculate the reliability of a context word we used the same technique as Lowe (2000) and Lowe and McDonald (2000). The corpus was first dived into four equal sections and semantic spaces for each section were created using the same parameters as in the previous sampling method. For each context word we then conducted an ANOVA test (Analysis of Variance) across the column vectors in each of the four semantic spaces. The results of the ANOVA test gives us a p value for the level that there is a significant variance between the four columns of the context word. If p is high then the columns are not significantly different, therefore the context word has a reliable behaviour across the corpus. However, if p is low then the columns are significantly different and the context word is inconstantly used across the corpus. Thereby the context words were sorted by those with the highest p values to the lowest p values in order to obtain a list of context words sorted by reliability.

The functions used to sample the basis can be found in Appendix B.3. After obtaining the

sets of sorted context words, the test to find the optimal basis sample was performed.

Experimental Setup

A semantic space was created for each combination of the following parameters:

- Basis sample type: most frequent, most variant and most reliable.
- Basis sample size: first n context words where $50 \leq n \leq$ total sample size.

Where each space had the following fixed features:

- Association Function: Flat window of size 8 and the log odds ratio.
- Similarity Metric: Cosine.
- Mapping: All k dimensions (no dimensionality reduction)

Each semantic space was then ran on the free association test and accuracy percentages were obtained.

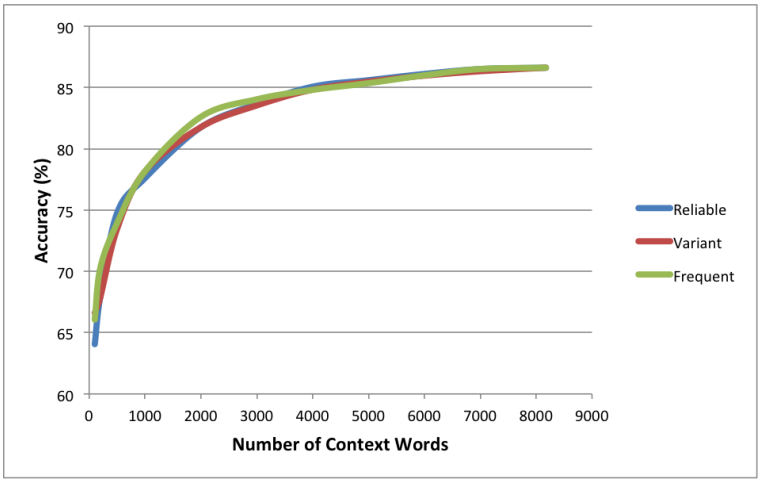

Figure 6.2: Accuracy results from the basis test. Graph shows the percentage of free associated words that were found for different semantic spaces when the sample type and sample size of the basis was varied

Results and Analysis

Figure 6.2 shows the results from the basis test where the optimum parameters we found to be:

- Sample of any type; with
- Sample size of 8175 context words; giving
- Accuracy of 86.62%

As can be seen from Figure 6.2 there seems to be no difference in the accuracy of the system when the sampling method is varied. Therefore there is no gain over using the most reliable or variable context words over the most frequent. Instead, the results seem to suggest the number of components (context words) in our space is the biggest contributing factor to the performance of the space. As a large proportion of the target words in the space occur very infrequently (see Zipf's Law, Section 4.2.1), as many context words must be utilised in order to obtain a useful amount of co-occurrence values for infrequent target words; this is consistent with results found by Bullinaria and Levy (2007). Furthermore we are unable to observe the decrease in performance when including infrequent, invariant and unreliable context words that we predicted in Section 6.1.4. One reason for this could be that by discarding context words that occur less than 50 times, the majority of the unreliable context words have already been removed which could pose a threat to the accuracy of our system. Moreover, the lack of variance between the three sampling techniques seems to suggest that by sorting by the most reliable or the most variable context words, they are still biased by frequency. This suggests that there is still work to be done to investigate the relationship between these different sampling methods.

The results show that using all context words that have a frequency value ≥ 50 gives us the best performance, therefore this was used in the following tests as the basis.

6.2.5 Similarity Metric Test

As identified in Section 6.1.4, due to inconstant findings, a similarity metric test was conducted to find whether the Euclidean distance, Cosine, City Block, Hellinger distance, or Kullback-Liebler divergence is the most appropriate for our model. As the Euclidean distance and City Block are distance functions; smaller values produced by these functions relate to high similarity. Conversely, with Cosine, Hellinger and Kullback-Liebler, larger values represent higher similarity between two words.

Experimental Setup

A semantic space was created for each of the following parameters:

- Similarity Metric: Cosine, Euclidean, City Block, Hellinger, Kullback-Liebler

Where each sematic space had the following fixed parameters:

- Association Function: Flat window of size 8 and the log odds ratio.

- Basis: All context words with frequency value ≥ 50.
- Mapping: All k dimensions (no dimensionality reduction).

Each space was then ran on the free association test and accuracy percentages were obtained.

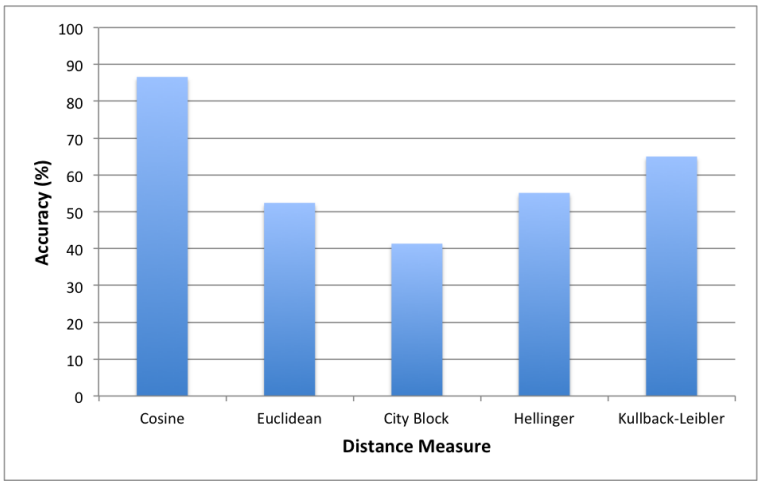

Figure 6.3: Accuracy results from the similarity metric test. Graph shows the percentage of free associated words that were found for different semantic spaces with the metrics: Cosine, Euclidean, City Block, Hellinger, Kullback-Liebler

Results and Analysis

Figure 6.3 shows the results from the similarity metric test where the optimal parameter was found to be:

- Cosine similarity measure; giving
- Accuracy of 86.62%

All other similarity metrics produce a significant decrease in performance with Kullback-Liebler providing an accuracy of 65.01%, Hellinger with an accuracy of 55.15%, Euclidean with an accuracy of 52.43% and City Block with an accuracy of 41.35%. For this reason the Cosine was selected as the optimal similarity metric for our semantic space and used in the final mapping test.

6.2.6 Mapping Test

The final test was conducted was to see whether any improvements could be made to our semantic space by reducing the dimensionality using SVD. The Python library called Scipy, includes an implementation of the LAPACK algorithm for reconstructing a matrix using SVD into the form $M = U\Sigma V^T$ (as described in Section 4.3.1). Similarly to Landauer and Dumais (1997), to reduce the dimensional space of the Enron Semantic Space to k dimensions, all but the k highest values in Σ were set to 0.

Experimental Setup

We created a semantic space for each of the following parameters:

- Mapping: 100$\geq k \geq$8199 (all dimensions)

Where each semantic space had the following fixed parameters:

- Association Function: Flat window of size 8 and the log odds ratio
- Basis: All context words with frequency value ≥ 50
- Similarity Metric: Cosine

As well as calculating the accuracy of each space on the free association test (shown in Figure 6.4) the average difference between the similarity of a target word and its associated word compared to each <*target word,unrelated word*> was calculated (shown in Figure 6.5).

Results and Analysis

Figure 6.4 shows the accuracy results of the mapping test. Although a significant improvement was not noted as hoped, an increase in accuracy can be seen where:

- k = 600; giving
- Accuracy = 86.86%

The significant drop in performance after 600 dimensions suggests there is too much information lost at a lower dimensional space. Interestingly, this dimensional threshold is much greater than the point found by Landauer and Dumais (1997), who found that there was a significant reduction in performance after 300 dimensions. Landauer et al. (1998) found that there was a linear increase in performance up to the threshold and a linear decrease

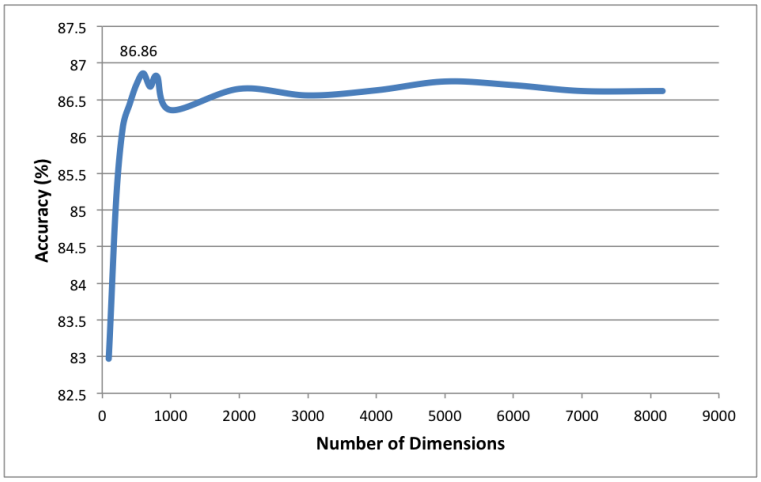

Figure 6.4: Accuracy results from the mapping test. Graph shows the percentage of free associated words that were found for different semantic spaces when the dimensionality of the space was varied.

after. Unfortunately this could not be reproduced this as there is a non linear behaviour between dimensions 8199 to 600 and only a linear decrease after. One explanation for this could be that the Enron corpus is in insufficient to observe any greater improvement in performance due to the words not being used in enough variety of contexts.

Therefore, as well as noting how often our semantic space predicts that an associated word is more associated than an unrelated word, we also analysed, on average, how similar the associated word is valued compared to unrelated words. The difference in similarity is found by subtracting the Cosine of the <*target word*,*associated word*> pair by the <*target word*, *unrelated word*> pair. Figure 6.5 shows this difference in similarity where positive values indicate the amount that associated words are viewed as being more similar than unrelated words (remembering that the range of Cosine values for SVD is [-1,1]). Now it can be observed how SVD has a dramatic affect on our semantic space. Even though SVD does not significantly increase the number of associated words found, in increases the similarity value of associated words and decreases the similarity of unrelated words.

Our system must be able to find as many associated pairs as possible whilst also finding a significant difference between the associated pairs and unrelated pairs. For this reason, we selected a dimensionality reduction of 600 dimensions as it provides an acceptable similarity difference value of 0.149, whilst also improving the accuracy of finding associated words.

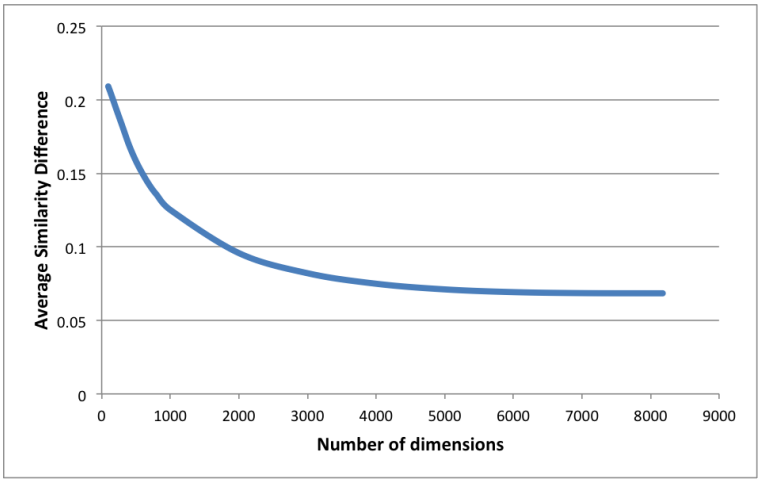

Figure 6.5: Similarity difference results from the mapping test. Graph shows the average difference between similarity ratings for associated and unrelated words when the dimensionality of the space was varied.

6.2.7 Optimum Implementation

The empirical analysis has thus given us the optimal implementation for observing direct cognitive priming:

- Association Function: Flat window of size 8 and the log odds ratio.
- Basis: All context words with frequency value ≥ 50.
- Similarity Metric: Cosine.
- Mapping: Reduce dimensional space to 600 dimensions using SVD.

6.3 Validation of Enron Semantic Space

The final requirement of our semantic space is that, as well as observing direct cognitive priming, it must also be able to observe graded and mediated priming by reproducing the studies carried out by McKoon et al. (1992) and Balota and Lorch (1986). In this section the results from the experiments that attempted to replicate those studies are presented.

6.3.1 Reproducing Graded Priming

As described in Section 3.1.2, *graded priming* is the phenomenon where strongly associated words produce better priming affects than weakly associated words. We remind the reader

that McKoon et al. (1992) sorted primes into four categories by strength of association. Ordered from strongly associated to not associated these are: free-associates, high t-primes, low t-primes and unrelated words. In their experiment, they found that free-associates produced the best recognition times when priming a target word. This was followed by high t-primes, then low t-primes and finally unrelated words produced the worst recognition times. For our system to observe graded priming it must be able to replicate similar results using the same stimulus used in their experiment. Rather than observing recognition times, the Enron Semantic Space, must be able to find on average that free-associates produce the best similarity rating. This should be followed by high t-primes, then low t-primes and finally with unrelated words producing the worst similarity rating. The graded priming experiment which was conducted to observe this behaviour will now be described along with the results obtained.

Experimental Setup

- *Stimulus*:
  - We obtained the set of target words, free-associates, high t-prime and low t-primes used by McKoon et al. (1992). From each set we removed any words that were not included in the set of target words for the Enron Semantic Space (described in Section 6.1.1), and therefore those that occurred less than 50 times in the Enron corpus. These quadruples can be found in Appendix A.1.
  - For each quadruple <*target word*,*free-associate*,*high t-prime*,*low t-prime*> we also randomly selected four other words from other quadruples to represent words unrelated to the target word.
- Semantic space: Optimum implementation of the Enron Semantic Space described in Section 6.2.7.

Using the semantic space, we calculated average the similarity values for <*target word*,*free-associate*>, <*target word*,*high t-prime*>, <*target word*,*low t-prime*> and <*target word*,*unrelated word*> pairs.

Results and Analysis

Figure 6.2 shows the comparison of our similarity ratings to the recognition times found by McKoon et al. (1992) for each of the priming groups. In this table lower reaction times and higher Cosine values represent stronger association.

As we hoped, free-associates gave the highest similarity values, followed by high t-primes, then low t-primes and lastly unrelated words. Using an ANOVA test it was also found that

	Free Associate	High T-Prime	Low T-Prime	Unrelated
McKoon et al. (1992) (RT in ms)	500	528	532	549
Enron Semantic Space (Cosine)	0.357	0.289	0.254	0.194

Table 6.1: Comparison between graded priming results found by McKoon et al. (1992) and the graded priming test performed on the Enron Semantic Space

free-associate similarities were significantly different to unrelated words ($F(2,31)=31.003, p<0.001$), high t-primes were significantly different to unrelated words ($F(2,31)=24.463, p<0.001$) and low t-primes were significantly different to unrelated words ($F(2,31)=8.25, p<0.01$). Furthermore the results were also significant across all groups ($F(4,31)=12.08$, $p<0.001$). So in conclusion, not only can it be verified that our system can observe graded priming, but the results have been obtained are statistically reliable.

6.3.2 Reproducing Mediated Priming

In Section 3.1.3, another type of priming called *mediated priming* was introduced. A mediated prime is a word that is indirectly associated to the target word through a mutual association to another word (for example lion and stripes are both associated to tiger, so lion is a mediated prime of stripes and vise versa). Balota and Lorch (1986) supported this theory of mediated priming by showing on a pronunciation task that mediated primes produced higher reaction times than unrelated words, but lower reaction times than direct primes. In order for our system to observe mediated priming, using the same stimulus, it must have the ability to find that direct primes are more similar to the target words than mediated primes and mediated primes are more similar than unrelated words. The mediated priming experiment that was conducted in order to observe this behaviour will now be described along with the results obtained.

Experimental Setup

- *Stimulus*:
  - We obtained the set of target words, direct primes and mediated primes used by McKoon et al. (1992) . From each set any words that were not included in the set of target words for the Enron Semantic Space (described in Section 6.1.1) were removed.
  - For each triple <*target word*,*direct prime*,*mediated prime*> four other words from other quadruples were randomly selected to represent words unrelated to the target word

- Semantic space: Optimum implementation of the Enron Semantic Space described in Section 6.2.7.

Using the semantic space, the average the similarity values for <*target word*,*direct prime*>, <*target word*,*meditated prime*> and <*target word*,*unrelated word*> pairs were calculated.

	Direct Prime	Mediated Prime	Unrelated
Balota and Lorch (1986) (RT in ms)	527	567	574
Enron Semantic Space (Cosine)	0.330	0.209	0.170

Table 6.2: Comparison between mediated priming results found by Balota and Lorch (1986) and the mediated priming test performed on the Enron Semantic Space

Results and Analysis

Figure 6.2 shows the comparison of our similarity ratings to the recognition times found by Balota and Lorch (1986) for each of the priming groups. Similar to before, lower reaction times and higher Cosine values represent stronger association.

As hoped, direct primes gave the highest similarity values on average, followed by meditated primes and lastly unrelated words. Using an ANOVA test in a similar fashion to before it was found that direct prime similarities were significantly different to unrelated words ($F(3,36)=55.849$, $p<0.001$), mediated primes were significantly different to unrelated words ($F(3,36)=7.061$, $p<0.01$) and there was a significant difference across all groups ($F(3,36)=33.153$, $p<0.001$). Thereby we can conclude that not only does the Enron Semantic Space observe mediated priming but the results obtained are statistically reliable.

6.4 Chapter Discussion

In this chapter the Enron Semantic Space was presented as a model of word use in the Enron Corpus. In our design multiple features from HAL, LMS and LSA were incorporated. A free association test was created to judge the performance of different parameters in our model and their effectiveness in the ability to accurately reproduce direct priming. Using this test an optimal implementation was found that predominately supported features from

Lowe and McDonald (2000)'s space over HAL, and gave a very successful score of 86.86% association accuracy. Furthermore, we were able to show for the first time that SVD can be used to improve accuracy of direct priming and can significantly increase the similarity difference between associated words and unrelated words. Our optimal implementation was validated by replicating both graded and mediated priming effects. As our model observes all types of cognitive priming one can be confident that our system will be able to observe associations between concepts and attributes in the same way that the IAT does. In the next chapter we present our application that uses the Enron Semantic Space to observe cultural attitudes and stereotypes upheld by the Enron Corporation. Exploratory experiments were then performed to demonstrate the use of semantic spaces as an indirect measure of attitudes.

Chapter 7

Indirectly Measuring Attitudes

As the Enron Semantic Space successfully observes cognitive priming, it is possible for us to switch the focus to examining whether this model can indeed be used as an alternative measure to the Implicit Association Test. In the discovery of attitudes, two different groups of 'concept' words are compared with their similarity to two different groups of 'attribute' words; thereby replicating the experimental procedure discussed in Section 2.1. To assist us in the exploration of these attitudes, an application was designed that allows the user to perform such a test. The first section of this chapter we describes this application, before proceeding to use it to perform indirect assessments of attitudes.

7.1 Enron Semantic Space Application

Continuing the tradition from the previous chapters Python and specifically the Tkinter libraries were used in order to design a graphical user interface for the Enron Semantic Space Application. This application allows the user to create an Enron Semantic Space of their own, with parameter combinations they desire, this can then be used to perform attitude analysis. In order to use this application, the user is required to have obtained the pre-processed corpus discussed in Chapter 5. This then needs to be hosted on a MySQL server and associated with a user account and password, so that the application can gain access to the Enron Corpus.

The initial screen on startup of the application is shown in Figure 7.1, where the functionality of the system it restricted until the user either loads a previously created semantic space file (ending in ".espace") or creates a new semantic space from scratch. By clicking on the *Create Semantic Space* button, the user is directed to a connection setup screen shown in Figure 7.2. Before the user can specify the parameters for creating the semantic space they must first connect to the Enron Corpus that they have hosted. In order to connect to the Enron Corpus they must provide the host IP address (for example "localhost" if on the same system as the application), the name of the database, the user account associated

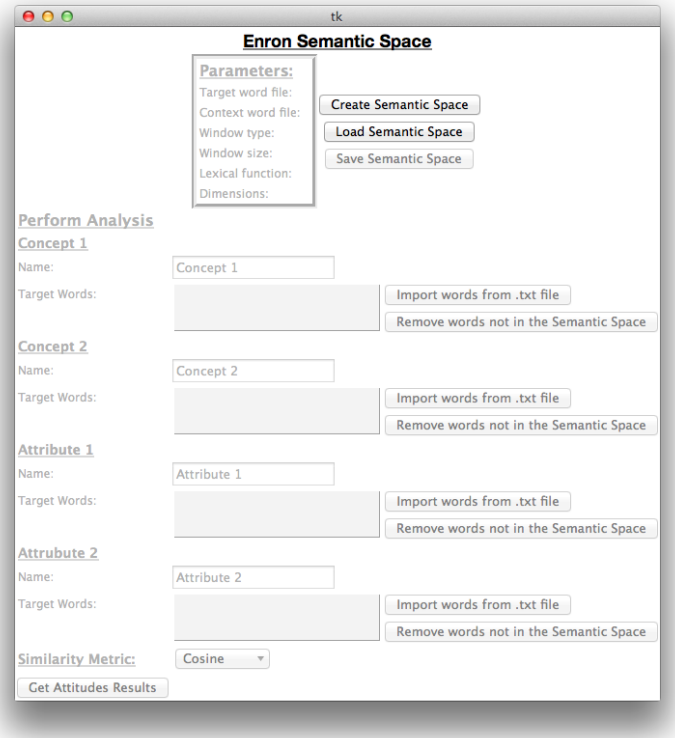

Figure 7.1: Enron Semantic Space application: Initial screen on startup

with the database and the user's password. If the information received is correct, then the application proceeds onto the model parameter screen shown in Figure 7.3.

In this screen the user can import their own set of context words and target words from a text file. Furthermore they can specify parameters that they wish to incorporate in to their semantic space from the options discussed in Section 6.1.4. The values for the *window type*, *window size*, *lexical function* and *dimensions* are set by default to those found in the optimal implementation (see 6.2.7). After submitting their choice of parameters, the program proceeds to create the semantic space using the steps mentioned in Section 6.1.5. As this is a time-consuming affair, taking on average 20 minutes to complete, the application displays a window showing how the system is progressing through the analysis of the corpus (see Figure 7.4).

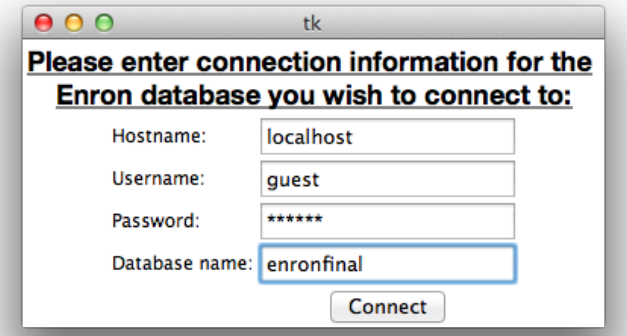

Figure 7.2: Enron Semantic Space application: Connection screen

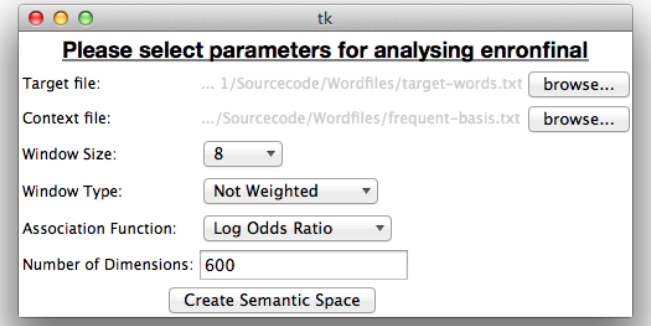

Figure 7.3: Enron Semantic Space application: Model parameter screen

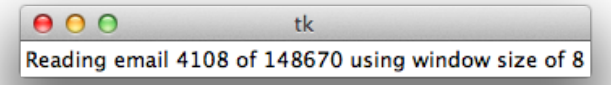

Figure 7.4: Enron Semantic Space application: Creation progress screen

After the semantic space has been created the rest of the functionality of the system is enabled for attitude analysis, which can be seen in Figure 7.5. If the user wishes, they can save the semantic space as an ".espace" file. When using the full set of target words in Section 6.1.1 and context words in Section 6.1.2, a file of around 1.5GB in size is produced. Although this file size is extremely large, this disadvantage is outweighed by increase in efficiency obtained by not having to create the semantic space every time the application is used. Rather than the 20 minutes that it takes to create a semantic space, it takes less than 5 minutes for the system to load a pre-existing model. The box at the top of the screen displays the parameters that were used to create the semantic space, so that if a user creates multiple semantic spaces with different parameters then they will be able to distinguish between them.

The *Perform Analysis* section of the screen is where the user can specify the experimental material to be used in the attitude analysis. To perform the analysis the user is required to enter two different sets of *concept* words (in *Concept 1* and *Concept 2*) and two different sets of *attribute* words (in *Attribute1* and *Attribute2*) . On submission of the words the system calculates:

- The average similarity of all words in Concept 1 with all words in Attribute 1
- The average similarity of all words in Concept 2 with all words in Attribute 1
- The average similarity of all words in Concept 1 with all words in Attribute 2
- The average similarity of all words in Concept 2 with all words in Attribute 2

Word sets can either be typed into the text boxes by hand or imported from a text file. As we wanted the application to be reasonably flexible, any word is allowed to be contained in the word sets. However any words that are not also in the set of target words used in creating the semantic space will give a similarity rating of 0, therefore a button was also created that automatically removes any of these words. Words that are removed are displayed to the user and if no words have been removed then the user is also notified.

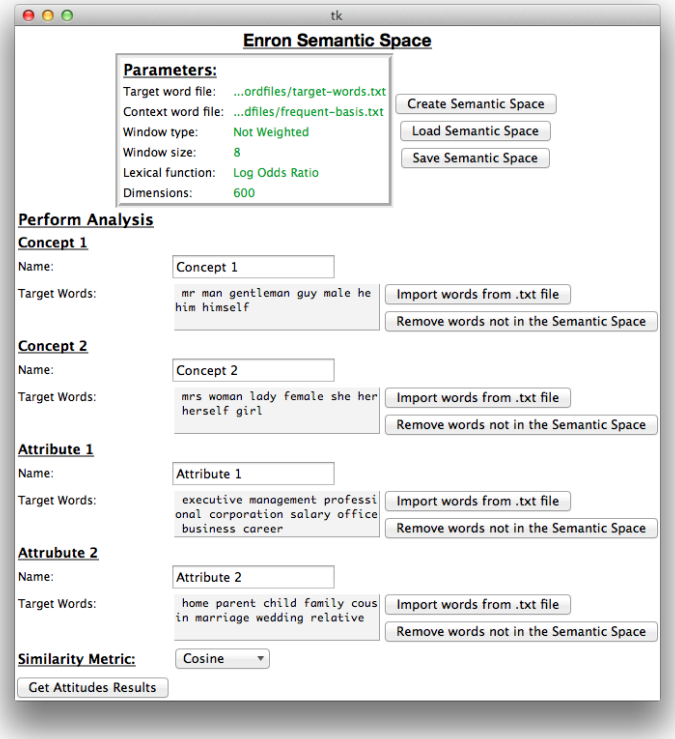

Figure 7.5: Enron Semantic Space application: Main screen after semantic space has been loaded or created

In the comparison of the words, the user can also decide which similarity metric to use, although we have set the default to Cosine as this was found to be optimal in Section 6.2.7.

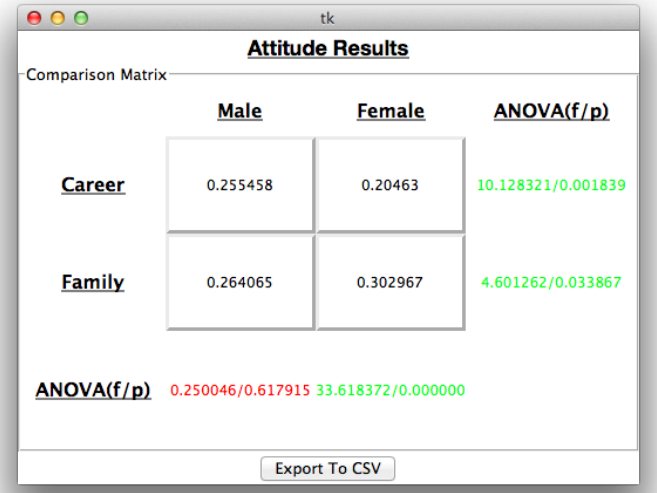

Figure 7.6: Enron Semantic Space: Attitude comparison results screen

Finally, once the user has submitted the experimental stimulus for assessment the application displays the average similarity values in a comparison matrix shown in Figure 7.6. As well as the average similarity values, the matrix also displays the results of ANOVA tests within groups where ANOVA(f/p) corresponds with the f and p value observed by the ANOVA test. For example, the ANOVA(f/p) that is found in the last value of the *Male* column calculates whether the *Male/Career* similarities are significantly different to the *Male/Family* similarities. Equally, the ANOVA(f/p) at the end of the *Career* row distinguishes whether *Male/Career* similarities are significantly different to the *Female/Career* similarities. In most statistical tests any p values that are less than 0.05 are said show a significant difference. We indicate to the user whether there is a significant difference within groups by changing the font colour to green the if $p<0.05$ (significant and reliable) or to red if $p \geq 0.05$ (insignificant and unreliable). Furthermore, the results obtained can also be exported as a CSV file for further manipulation.

7.2 Exploring Attitudes in the Enron Semantic Space

In a perfect scenario, one would run the same experimental stimuli from successful IAT's and attempt to reproduce similar results using the application created. However, we were

unable to discover a single experiment where all of the words that were used appeared more than 50 times in the Enron Corpus. We remind the reader that there are 8,199 target words in the Enron Corpus which is barely 2% of the total number words in the English vocabulary [1]. Therefore, rather than directly replicating those studies, previous IAT findings were utilised to conduct our own exploratory research. For all experiments the optimal parameters from 6.2.7 were used to create the semantic space.

7.2.1 Pleasantness Attitudes

In the first set of attitude tests, sets of concepts and their similarities to pleasant and unpleasant meaning words were compared.

Life vs Death

As mentioned in Section 2.1, the initial study conducted by Greenwald et al. (1988) was to verify that their IAT could observe near universally accepted attitudes. In their study they successfully found that flowers were strongly associated with pleasantness and insects were associated with unpleasantness. Inspired by this fundamental study, we created a universally accepted attitude test of our own. As sufficient occurrences of different types of flowers and insects could not be observed, two broader concepts were chosen instead; life and death. In doing so four sets of words we obtained:

- *Life concept words*: life, flower, plant, tree, growth, child, birth, baby, wildlife, begin, activity, energy, forest, animal
- *Death concept words*: death, die, gun, terrorist, weapon, shotgun, bomb, kill, murder, killer, end, destruction, destroy, suicide
- *Pleasant attribute words*: good, pleasant, nice, lovely, wonderful, fun, jolly, sweet, kind, agreeable
- *Unpleasant attribute words*: bad, awful, evil, horrible, disturbing, sick, sour, unpleasant, nasty, terrible

If the Enron Semantic Space is able to observe attitudes, then it should have the ability to find that words related to *life* are more similar to *pleasant* meaning words than *unpleasant* meaning words. Converseley, it should also find that words related to *death* are more similar to *unpleasant* meaning words than *pleasant* meaning words.

[1] The Oxford English Dictionary contains over 600,000 different word types (Simpson, Weiner et al., 1989)

	Life	Death	ANOVA (f/p)
Pleasant	0.20387	0.15342	21.63919/0.00000
Unpleasant	0.15163	0.16920	5.05745/0.02523
ANOVA (f/p)	22.97580/0.00000	4.50254/0.03468	

Table 7.1: Attitude comparison matrix for concepts *life* and *death* and attributes *pleasant* and *unpleasant*

Table 7.1 displays the results obtained using the Enron Semantic Space application and the life vs death stimuli. From this, the following reliable observations can be made:

1. *Life* words are more similar to *pleasant* words (0.20387), than *unpleasant* words (0.15163), with the ANOVA test giving p=0.00000.
2. *Death* words are more similar to *unpleasant* words (0.16920), than *pleasant* words (0.15342), with the ANOVA test giving p=0.03468.
3. *Pleasant* words are more similar to *life* words (0.20387), than *death* words (0.15342), with the ANOVA test giving p=0.00000.
4. *Unpleasant* words are more similar to *death* words (0.16920) than *life* words (0.15163), with the ANOVA test giving p=0.02523.

In conclusion, this study proves our system can observe universally accepted attitudes as life words were biased towards pleasantness and death words were biased towards unpleasantness. Subsequently, this also provides encouraging evidence that the words chosen to represent pleasantness and unpleasantness are appropriate, as pleasant is more similar to life than death and unpleasant is more similar to death than life.

Enron vs Dynergy

A large proportion of IAT research has revolved around observing in-group bias. This is where individuals have stronger positive attitudes towards their own social group than other social groups. Greenwald et al. (1988) observed this with phenomenon with Korean and Japanese students and Kühnen, Schießl, Bauer, Paulig, Pöhlmann, Schmidthals et al. (2001) found a similar relationship between East and West Germans. Furthermore, in-group bias has also been observed between Jewish and Christian participants (Rudman, Greenwald, Mellott and Schwartz, 1999). In the next step, in-group bias was taken to the next level by comparing positive attitudes between Enron and Dynergy, who were widely regarded as being one of Enron's major competitors (Healy and Palepu, 2003). Due to the preprocessing steps taken in Chapter 5 we can be confident that all the emails are from people working at Enron. Therefore, we might expect to see an in-group attitude bias for Enron over Dynergy. Furthermore, as Dynergy were competitors of Enron, we might also

expect to observe a negative attitude towards them. In the second experiment, the words *enron* and *dynergy* were compared with the pleasant and unpleasant words in the previous experiment.

	Enron	Dynergy	ANOVA (f/p)
Pleasant	0.16710	0.10673	8.40849/0.00955
Unpleasant	0.10585	0.13868	3.73619/0.06753
ANOVA (f/p)	7.63083/0.01240	4.65743/0.04393	

Table 7.2: Attitude comparison matrix for concepts *enron* and *dynergy* and attributes *pleasant* and *unpleasant*

Table 7.2 displays the results obtained using the Enron Semantic Space application and the Enron vs Dynergy stimuli. From, this we can extract the following reliable conclusions:

1. *Enron* is more similar to *pleasant* words (0.16710) than *unpleasant* words (0.10585), with the ANOVA test giving p=0.01240.
2. *Dynergy* is more more similar to *unpleasant* words (0.13868) than *pleasant* words (0.10673), with the ANOVA test giving p=0.04393.
3. *Pleasant* words are more similar to *Enron* (0.16710) than *Dynergy* (0.10673), with the ANOVA test giving p=0.00955.

Therefore an in-group bias of positive attitudes for Enron over Dynergy has been observed. Furthermore, it was also discovered that because Dynergy was a competitor, there is a negative attitude towards them. However, we could not observe a significant bias for unpleasantness to be more associated with Dynergy than Enron. One reason for this could be that Enron was used in many unpleasant contexts during the 'Enron scandal' which could have thereby increased its similarity rating for unpleasantness.

Elderly vs Youth

In a photo classification task, Hummert et al. (2002) compared the differences in association between photos of young and old people with pleasant and unpleasant objects. In this experiment, they found a significant bias of positive attitudes towards youthful photos over elderly photos. Using the same set of pleasant and unpleasant words as before, we hoped to find similar results when comparing the associations between elderly and youth concept words with pleasantness. In this experiment the following concept words were used:

- *Elderly concept words*: veteran, elderly, grandfather, grandma, grandmother, old
- *Youth concept words*: intern, youth, graduate, undergraduate, student, young

The words were chosen so that they had a neutral valence. Words such as 'immature' for youth was not included as it has a negative meaning, conversely, 'mature' for elderly was not included as it has a positive meaning.

	Elderly	Youth	ANOVA (f/p)
Pleasant	0.21334	0.18222	3.33395/0.07039
Unpleasant	0.17135	0.11282	23.93334/0.00000
ANOVA (f/p)	5.99070/0.01578	37.93837/0.00000	

Table 7.3: Attitude comparison matrix for concepts *elderly* and *youth* and attributes *pleasant* and *unpleasant*

Table 7.3 displays the results obtained using the Enron Semantic Space application and the elderly vs youth stimuli. The results did not produce the attitude bias that we hoped to observe as elderly was found to be more similar to both pleasantness and unpleasantness compared to youth. The reason for this could be because words such as *grandfather*, *grandma* and *grandmother* were included which refer relatives. Therefore they will predominately be used in positive contexts. Although this test did not provide successful results, we continued to try to observe an age bias by comparing competence attitudes towards elderly and youth.

7.2.2 Age Competence Attitudes

Kite, Stockdale, Whitley and Johnson (2005) conducted a meta-analysis review of numerous psychological studies into ageism. One of the dimensions of ageism studied compared competence ratings for young and elderly candidates. Subsequently, they found a significant preference for young participants to be given a higher competence rating than elderly candidates.

In the next test we endeavoured to see whether the attitude that youth is more associated with competence than elderly can be observed. To do this, the same youth and elderly words used in 7.2.1 and their similarity ratings to the following competent and incompetent attribute words were compared:

- *Competent attribute words*: reliable, successful, intelligent, smart, responsible, good, acceptable, useful, helpful, valuable, aid, benefit, capable, adequate
- *Incompetent attribute words*: unreliable, unsuccessful, stupid, dumb, irresponsible, bad, unacceptable, useless, detrimental, worthless, burden, handicap, unable, inadequate

	Elderly	Youth	ANOVA (f/p)
Competent	0.14421	0.17502	8.55443/0.00393
Incompetent	0.14919	0.11490	15.38940/0.00013
ANOVA (f/p)	0.20133/0.65423	56.35491/0.00000	

Table 7.4: Attitude comparison matrix for concepts *elderly* and *youth* and attributes *competent* and *incompetent*

Table 7.4 displays the results obtained using the Enron Semantic Space application and the elderly vs youth and competent vs incompetent stimuli. From this we can observe the following:

1. *Competent* words are more similar to *youth* concept words (0.17502) than *elderly* concept words (0.14421), with ANOVA giving p=0.00393.
2. *Incompetent* words are more similar to *elderly* concept words (0.14919) than *youth* concept words (0.11490), with ANOVA giving p=0.00013.

Subsequently, we can conclude that there is an attitude bias towards youth and competence, and elderly and incompetence, which is consistent with Kite et al. (2005).

7.2.3 Gender Stereotyping

In our final experiment, we tried to ascertain as to whether there are any gender stereotypes upheld by the Enron Corporation. As mentioned in Section 2.2, Nosek et al. (2002) observed the bias towards associating men with the notion of career and women with notion of family. In their IAT test they compared the association of male and female names with words related to career and family. The final test attempts to observe similar associations, however instead of using male and female words, words related male and female concepts were used. The reason for this is that names predominantly occur at the beginning and end of emails so useful information into their use would not be obtained. The following words were used as experimental stimuli:

1. *Male words*: mr, man, gentleman, guy, male, he, him, himself
2. *Female words*: mrs, woman, lady, female, she, her, herself, girl
3. *Career words*: executive, management, professional, corporation, salary, office, business, career
4. *Family words*: home, parent, child, family, cousin, marriage, wedding, relative

Most of the words used in the *career* and *family* set were taken directly from Nosek et al. (2002), however plurals such as children, cousins and relatives were converted into their

singular form due to the lemmatisation done in Section 5.2.4. Furthermore gender words such as *wife*, *husband*, *mother*, *father* were not included as they would be predominantly biased towards the family notion.

	Male	Female	ANOVA (f/p)
Career	0.25546	0.2046	10.12832/0.00184
Family	0.264065	0.302967	4.601262/0.033867
ANOVA (f/p)	0.25005/0.61792	33.61837/0.00000	

Table 7.5: Attitude comparison matrix for concepts *male* and *female* and notions *career* and *family*

Table 7.5 displays the results obtained using the Enron Semantic Space application and the elderly vs youth and competent vs incompetent stimuli. From this we can observe the following:

1. *Career* words are more similar to *male* concept words (0.25546) than *female* concept words (0.2046), with the ANOVA test giving p=0.00184.
2. *Family* words are more similar to *female* concept words (0.302967) than *male* concept words (0.264065), with the ANOVA test giving p=0.033867.

Consequently, we have supported findings by Nosek et al. (2002) in that the Enron Corporation also has a bias towards associating male with career and female with family.

7.3 Chapter Disscussion

In this chapter we have presented the application created in order to indirectly observe attitudes in the Enron Corpus. After creating this application experiments were undergone which demonstrated the use of semantic spaces as an indirect measure of attitudes. In those experiments universal attitudes that life is viewed as being pleasant and death is viewed as being unpleasant were successfully observed. An in-group bias for preferring Enron over their competitor Dynergy and that there is negative attitude bias towards Dynergy was also observed. By comparing age with competence, the study done by Kite et al. (2005) was also supported by finding that youth was associated with competence and elderly associated with incompetence. Finally, gender stereotyping was also found where male was more associated career and female was associated with family.

Chapter 8

Conclusions

The fear of expressing beliefs that are contrary to social norms suggests that only way attitudes can be reliably observed is through indirect measures. The Implicit Association Test (IAT) is a popular example of such a measure which observes automatic associations between concepts and attributes without the subject realising that an attitude is being assessed. Although the IAT has been able to observe universally held attributes and interesting racial, gender and age related biases, it is exposed to extraneous influences that are difficult to completely control for. Alternatively to the IAT, automatic associations could be observed by analysing the similarities in the use of words in natural language. By adopting word use as a measure of attitudes, we remove the involvement of participants completely and subsequently eliminate the extraneous influences.

An individual's word use can be modelled by a semantic space, where words are represented as an n-dimensional vector of the number of times they co-occur with neighbouring context words. By parsing the Enron Corpus, we have created the Enron Semantic Space of word use across emails sent from individuals employed by the Enron Corporation. In preparation for the semantic analysis the corpus has been pre-processed using techniques such as tokenisation, parts-of-speech tagging and lemmatisation. Furthermore negative uses of words have been annotated so the semantic space does not incorporate its opposite meaning into the model. We have shown that this pre-processed corpus contains a rich amount of useful information, that could be used for a number of different natural language processing tasks.

A semantic space that attempts to observe automatic associations, needs to be able to reproduce the cognitive priming affects that are facilitated by the Implicit Association Test. Although there have been many different implementations of semantic spaces such as the Hyperspace Analogue to Language, Latent Semantic Analysis and Lowe and McDonald's Semantic Space (Lowe and McDonald, 2000), there are limited findings into the best implementation that can accurately observe all types of cognitive priming: direct, graded and mediated. A free association test was created, using the Univerity of Florida free association norms, which calculates the accuracy that a semantic space can distinguish between

directly associated words and unrelated words. Using this test we have found that the optimal combination of parameters coincide with LMS and LSA, rather than those specified by HAL. This implementation uses a flat window technique to capture the co-occurrences of target and context words, which are then converted into association values using the log odds ratio. Contrary to Lund and Burgess (1996) and Lowe and McDonald (2000) it was discovered that as many context words as possible need to be considered in order to accumulate the greatest amount co-occurrences for the word vectors. Furthermore, it has been found for the first time that the accuracy of the system in observing direct priming can be improved by reducing the dimensions of the model using Singular Value Decomposition. Subsequently, it was discovered that the Cosine similarity metric produced significantly better results than any other metric which compliments the study conducted by Bullinaria and Levy (2007). To validate the final parameters, the graded and mediated priming results obtained by McKoon et al. (1992) and Balota and Lorch (1986) were also replicated.

To analyse cultural attitudes in the Enron Semantic Space, an application has been created using Python's Tkinter libraries. This application provides a graphical user interface where the user can create new semantic spaces from the Enron corpus and which can be saved to disk for subsequent analysis. The application then allows the user to calculate the similarities between sets of concept and attribute words in the semantic space and therefore observe automatic associations in similar way to the IAT. This application was then used to conduct an indirect assessment of attitudes.

Experiments found that *life* concept words were associated with *pleasant* words and *death* concept words were associated with *unpleasant* words. This has allowed us to deduce that our system can observe the universal attitude that *life* is a *pleasant* concept and *death* is an *unpleasant* concept. Similarly, an in-group attitude bias towards *Enron* being related to *pleasantness* over their competitor *Dynergy* was found. Moreover *Dynergy* was also regarded as being more associated with *unpleasantness* than *pleasantness*.

Competence and *incompetence* was then used as an alternative evaluation of positive and negative attitudes. Similarly to Kite et al. (2005), it was observed that *youth* was related to *competence*, whereas *elderly* was related to *incompetence*. From this information we can infer that *youthfulness* is valued over experience in the culture of Enron. In a further experiment, there was tendency to adhere to the gender stereotype that *men* are more associated with *career* than *women*, and *women* are more associated with *family* than *men*.

In conclusion, the successful results obtained using the Enron Semantic Space help to contribute towards the use of semantic spaces as an indirect measure of attitudes. Nevertheless, the Enron corpus remains fairly inflexible due to the small number of words it contains. Therefore, it has been impossible to fully replicate findings from the Implicit Association Test. Before one can fully support the exchangeability of semantic analysis and the IAT, one must analyse multiple instances of larger corpora and find a consistent correlation with IAT results. If this can be achieved, then progression of attitude research could be quickened tenfold. Attitude research conducted by the IAT is limited by the num-

ber of participants that volunteer to take part in the experiments and time spent designing IAT's that cope with extraneous influences. However, if word use is a good measure of attitudes then the amount of materials available to study is limitless. Due to the nature of the communication age of today, samples of written language are freely available through social media everyday from individuals across the globe. Furthermore, if a semantic space can be designed that models the use of all words in our vocabulary, no more work needs to be done other than incrementally training this model with new information daily. Finally, if attitudes can predict one's behaviour then it might be possible to use semantic analysis to predict voting behaviour and consumerism from multiple communities in social networks.

Chapter 9

Future Work

In this final chapter three possible extensions in the semantic analysis of the Enron Corpus are considered: alternative definitions of context, alternative approaches to the mapping function and further studies of attitudes.

9.1 Alternative Definitions of Context

In the design of the Enron Semantic Space, the definition of a word was restricted to the way in which it co-occurs with other words. However, this is not the only way that the Enron Semantic Space could have defined word use. As mentioned in Section 6.1.2, emails weren't used as a definition of context due to the size of the matrix that would be required. However De Lathauwer, De Moor and Vandewalle (2000) have designed an extension to SVD, called Multi-linear Singular Value Decomposition, where the SVD of a distributed matrix can be computed. By using this algorithm the co-occurrence matrix could be distributed across memory units in a cluster and computed in parallel. This would allow the semantic space to scale to much larger matrices such as one generated by using emails as context.

Another way that a word could be defined is the way in which it is similar to other words. Such a semantic space would therefore be extremely similar to the spreading activation theory of semantic memory discussed in Section 3.2.1. The more words that two words are both associated with, the more similar that are. Using SVD one can only speculate that mediated priming effects are observed by replicating psychological experiments. However by using similarity to all other words as the context, mediated priming is directly implemented. If two words are indirectly related to each other then they will be valued as being more similar than two words that are neither directly or indirectly related.

A novel way in which words could be defined in the semantic space, is by people that use them. The Enron Corpus only contains emails from around 150 employees (see Section

5), so a corpus that contains far more individual authors would need to be analysed so that the number of coefficients in the space is significant. Homophily is the observation made in sociology where people are more likely to form relationships with those that they are similar to (McPherson, Smith-Lovin and Cook, 2001). One of the traits in which two people could be similar is in the way in which they prefer the use of certain words in the English vocabulary. Subsequently, if people can be defined by their word use then possibly words could be defined by the people that use them. An extension to the project could therefore analyse which definition of context best describes a word vector.

9.2 Other Approaches to the Mapping Function

One of the main contributions of the Enron Semantic Space is in showing that the accuracy of direct cognitive priming can be improved by mapping the semantic space onto another of reduced dimensionality (using SVD). Now that an improvement has been found using such a mapping function, other approaches could be implemented.

One alternative approach is Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 2001). PLSA maps the co-occurrence matrix to a space where the probability or the association of a word and a context is a mixture of conditional probabilities with a latent variable z. It calculates these probabilities using an Expectation-Maximisation function which approximates the representation of the semantic space in a series of incremental steps. PLSA is said to be able to observe polysemy (multiple meanings of a word) far more effectively than SVD. For this reason it may be a better model to observe multiple uses of words.

Latent Dirichlet Allocation (LDA) Blei, Ng and Jordan (2003) is another mapping approach for semantic spaces which define the context of a word by the documents that it appears in. LDA maps the co-occurrence matrix to a space where the association of a word with a document is defined over a number of unobserved topics that they share. An example of a topic could be *football related* which would link the word *goal* with some probability to a document about *football* even if *goal* never actually occurred within the document. This could easily be extended to the Enron Semantic Space were the context is defined by words rather than documents.

9.3 Further Attitude Analysis of Enron

Although a wide range of attitude analysis has been conducted in Section 7.2, there are still a number of experiments that could be conducted on the Enron Semantic Space. One experiment could analyse the way in which attitudes change over time. As attitudes are affected by social experiences, any new experiences could completely change the attitudes that we hold. The Enron Corpus contains a record of word use between the 1999-2002, so

one could select different time periods of email activity and compare the cultural attitudes held at different times in the corpus. For example an extension to the Enron vs Dynergy experiment in Section 7.2, could try to identify whether attitudes towards Enron were different before and during the 'Enron Scandal'. It would interesting to find that Enron was more related to pleasant words before the 'Enron Scandal' and more related to unpleasant words during the 'Enron Scandal'. Furthermore it would also be interesting to see if the attitudes of people in Enron differed by age or gender. If one could obtain information on the ages of the authors of the emails, then we could divide emails sent by young and old employees and contrast their performance on the Youth vs Elderly competence experiment (Section 7.2.1). Similarly, if the emails could be separated by those sent by males and those sent by females we could also compare each set of emails and their results on the gender stereotyping experiment in Section 7.2.3.

Bibliography

Agresti, A. (1990), 'Models for matched pairs', *Categorical Data Analysis, Second Edition* pp. 409–454.

Ajzen, I. and Fishbein, M. (1977), 'Attitude-behavior relations: A theoretical analysis and review of empirical research.', *Psychological bulletin* **84**(5), 888.

Anderson, E., Bai, Z., Dongarra, J., Greenbaum, A., McKenney, A., Du Croz, J., Hammerling, S., Demmel, J., Bischof, C. and Sorensen, D. (1990), Lapack: A portable linear algebra library for high-performance computers, *in* 'Proceedings of the 1990 ACM/IEEE conference on Supercomputing', IEEE Computer Society Press, pp. 2–11.

Anderson, J. R. (1983), 'A spreading activation theory of memory', *Journal of verbal learning and verbal behavior* **22**(3), 261–295.

Andrew, F. and Heer, J. (2004), 'Enron email analysis'.
URL: *http://bailando.sims.berkeley.edu/enron_email.html*

Arcuri, L., Castelli, L., Galdi, S., Zogmaister, C. and Amadori, A. (2008), 'Predicting the vote: Implicit attitudes as predictors of the future behavior of decided and undecided voters', *Political Psychology* **29**(3), 369–387.

Aston, G. (1997), 'The bnc handbook exploring the british national corpus with sara guy aston and lou burnard'.

Balota, D. A. and Lorch, R. F. (1986), 'Depth of automatic spreading activation: Mediated priming effects in pronunciation but not in lexical decision', *Journal of Experimental Psychology: Learning, Memory, and Cognition* **12**(3), 336–345.

Bilovich, A. (2006), 'Detecting individual differences in beliefs through language use'.

Bilovich, A. and Bryson, J. J. (2008), Detecting the evolution of semantics and individual beliefs through statistical analysis of language use, *in* 'Naturally-Inspired Artificial Intelligence-Papers from the AAAI Fall Symposium'.

Blei, D. M., Ng, A. Y. and Jordan, M. I. (2003), 'Latent dirichlet allocation', *the Journal of machine Learning research* **3**, 993–1022.

Bullinaria, J. A. and Levy, J. P. (2007), 'Extracting semantic representations from word co-occurrence statistics: A computational study', *Behavior Research Methods* **39**(3), 510–526.

Burgess, C. (1998), 'From simple associations to the building blocks of language: Modeling meaning in memory with the hal model', *Behavior Research Methods, Instruments, & Computers* **30**(2), 188–198.

Church, K. W. and Hanks, P. (1990), 'Word association norms, mutual information, and lexicography', *Computational linguistics* **16**(1), 22–29.

Collins, A. M. and Loftus, E. F. (1975), 'A spreading-activation theory of semantic processing', *Psychological review* **82**(6), 407–428.

Constantine, M. G. and Ladany, N. (2000), 'Self-report multicultural counseling competence scales: Their relation to social desirability attitudes and multicultural case conceptualization ability.', *Journal of Counseling Psychology* **47**(2), 155.

Corver, N. and Van Riemsdijk, H. (2001), *Semi-lexical categories: the function of content words and the content of function words*, Walter de Gruyter.

Currall, S. and Epstein, M. (2003), 'The fragility of organizational trust: Lessons from the rise and fall of enron', *Organizational Dynamics* **32**(2), 193–206.

Danziger, P. (2004), *Why people buy things they don't need: understanding and predicting consumer behavior*, Kaplan Publishing.

De Houwer, J. (2006), 'What are implicit measures and why are we using them', *The handbook of implicit cognition and addiction* pp. 11–28.

De Lathauwer, L., De Moor, B. and Vandewalle, J. (2000), 'A multilinear singular value decomposition', *SIAM journal on Matrix Analysis and Applications* **21**(4), 1253–1278.

Dosher, B. A. and Rosedale, G. (1989), 'Integrated retrieval cues as a mechanism for priming in retrieval from memory', *Journal of experimental psychology. General* **118**(2), 191–211.

Dumais, S. (1991), 'Improving the retrieval of information from external sources', *Behavior Research Methods* **23**(2), 229–236.

Fano, R. (1961), 'Transmission of information'.

Fazio, R. H., Jackson, J. R., Dunton, B. C. and Williams, C. J. (1995), 'Attitudes and social cognition', *Journal of personality and social psychology* **69**(6), 1013–1027.

Fazio, R. H. and Olson, M. A. (2003), 'Implicit measures in social cognition research: Their meaning and use', *Annual review of psychology* **54**(1), 297–327.

Fishbein, M. and Ajzen, I. (2005), 'The influence of attitudes on behavior', *The handbook of attitudes* pp. 173–221.

Fox, L. (2002), *Enron: The rise and fall*, Wiley.

Francis, W. N. and Kucera, H. (1979), 'Brown corpus manual', *Letters to the Editor* **5**(2), 7.

Freud, S. (1938), 'Psychopathology of everyday life'.

Frost, R., Forster, K. I. and Deutsch, A. (1997), 'What can we learn from the morphology of hebrew? a masked-priming investigation of morphological representation', *JOURNAL OF EXPERIMENTAL PSYCHOLOGY LEARNING MEMORY AND COGNITION* **23**, 829–856.

Fusaro, P. and Miller, R. (2002), *What went wrong at Enron: Everyone's guide to the largest bankruptcy in US history*, Wiley.

Gillund, G., Shiffrin, R. M. et al. (1984), 'A retrieval model for both recognition and recall', *Psychological review* **91**(1), 1–67.

Golub, G. and Van Loan, C. (1996), *Matrix computations*, Vol. 3, Johns Hopkins University Press.

Greenwald, A. and Banaji, M. (1995), 'Implicit social cognition: attitudes, self-esteem, and stereotypes.', *Psychological review* **102**(1), 4.

Greenwald, A. G., Nosek, B. A., Banaji, M. R. et al. (2003), 'Understanding and using the implicit association test: I. an improved scoring algorithm', *Journal of personality and social psychology* **85**(2), 197–216.

Greenwald, A. G., Nosek, B. A. et al. (2001), 'Health of the implicit association test at age 3', *Zeitschrift für Experimentelle Psychologie* **48**(2), 85–93.

Greenwald, A., McGhee, D. and Schwartz, J. (1988), 'Measureing differences in implicit cognition: The implicit association test.', *Journal of Personality and Social Psychology* **74**, 1464–1480.

Halmos, P. R. (1947), *Finite dimensional vector spaces*, Vol. 7, Princeton University Press.

Hasan, F. M., UzZaman, N. and Khan, M. (2007), Comparison of different pos tagging techniques (n-gram, hmm and brill's tagger) for bangla, *in* 'Advances and Innovations in Systems, Computing Sciences and Software Engineering', Springer, pp. 121–126.

Healy, P. and Palepu, K. (2003), 'The fall of enron', *The Journal of Economic Perspectives* **17**(2), 3–26.

Hintzman, D. L. (1990), 'Human learning and memory: Connections and dissociations', *Annual Review of Psychology* **41**(1), 109–139.

Hofmann, T. (2001), 'Unsupervised learning by probabilistic latent semantic analysis', *Machine learning* **42**(1-2), 177–196.

Hofmann, W., Rauch, W. and Gawronski, B. (2007), 'And deplete us not into temptation: Automatic attitudes, dietary restraint, and self-regulatory resources as determinants of eating behavior', *Journal of Experimental Social Psychology* **43**(3), 497–504.

Hummel, J. E., Holyoak, K. J. et al. (2003), 'A symbolic-connectionist theory of relational inference and generalization', *Psychological review* **110**(2), 220–264.

Hummert, M. L., Garstka, T. A., O Brien, L. T., Greenwald, A. G., Mellott, D. S. et al. (2002), 'Using the implicit association test to measure age differences in implicit social cognitions', *Psychology and Aging* **17**(3), 482–495.

Jones, M. N., Kintsch, W. and Mewhort, D. J. (2006), 'High-dimensional semantic space accounts of priming', *Journal of memory and language* **55**(4), 534–552.

Kiss, T. and Strunk, J. (2006), 'Unsupervised multilingual sentence boundary detection', *Computational Linguistics* **32**(4), 485–525.

Kite, M. E., Stockdale, G. D., Whitley, B. E. and Johnson, B. T. (2005), 'Attitudes toward younger and older adults: An updated meta-analytic review', *Journal of social issues* **61**(2), 241–266.

Klimt, B. and Yang, Y. (2004), 'The enron corpus: A new dataset for email classification research', *Machine Learning: ECML 2004* pp. 217–226.

Kühnen, U., Schießl, M., Bauer, N., Paulig, N., Pöhlmann, C., Schmidthals, K. et al. (2001), 'How robust is the iat? measuring and manipulating implicit attitudes of east-and west-germans', *Zeitschrift für Experimentelle Psychologie* **48**(2), 135–144.

Lacan, J. and Wilden, A. (1968), *The language of the self: The function of language in psychoanalysis.*, Johns Hopkins University Press.

Landauer, T. and Dumais, S. (1997), 'A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.', *Psychological Review; Psychological Review* **104**(2), 211.

Landauer, T., Foltz, P. and Laham, D. (1998), 'An introduction to latent semantic analysis', *Discourse processes* **25**(2-3), 259–284.

Landy, D. and Sigall, H. (1974), 'Beauty is talent: Task evaluation as a function of the performer's physical attractiveness.', *Journal of Personality and Social Psychology* **29**(3), 299.

Levy, J. P., Bullinaria, J. A. and Patel, M. (1998), 'Explorations in the derivation of semantic representations from word co-occurrence statistics', *South Pacific Journal of Psychology* **10**(99), 111.

Lowe, W. (2000), Topographic maps of semantic space, PhD thesis, University of Edinburgh.

Lowe, W. and McDonald, S. (2000), The direct route: Mediated priming in semantic space, Technical report, The University of Edinburgh.

Lund, K. and Burgess, C. (1996), 'Producing high-dimensional semantic spaces from lexical co-occurrence', *Behavior Research Methods, Instruments, & Computers* **28**(2), 203–208.

Lund, K., Burgess, C. and Atchley, R. A. (1995), Semantic and associative priming in high-dimensional semantic space, *in* 'Proceedings of the 17th annual conference of the Cognitive Science Society', Vol. 17, pp. 660–665.

Maison, D., Greenwald, A. G. and Bruin, R. H. (2004), 'Predictive validity of the implicit association test in studies of brands, consumer attitudes, and behavior', *Journal of Consumer Psychology* **14**(4), 405–415.

Marcus, M. P., Marcinkiewicz, M. A. and Santorini, B. (1993), 'Building a large annotated corpus of english: The penn treebank', *Computational linguistics* **19**(2), 313–330.

Marslen-Wilson, W. D., Tyler, L. K. et al. (1997), 'Dissociating types of mental computation', *Nature* **387**(6633), 592–593.

Marslen-Wilson, W., Komisarjevsky Tyler, L., Waksler, R. and Older, L. (1994), 'Morphology and meaning in the english mental lexicon', *PSYCHOLOGICAL REVIEW-NEW YORK-* **101**, 3–3.

Marslen-Wilson, W. and Zhou, X. (1999), 'Abstractness, allomorphy, and lexical architecture', *Language and Cognitive Processes* **14**(4), 321–352.

McKoon, G., Ratcliff, R. et al. (1992), 'Spreading activation versus compound cue accounts of priming: Mediated priming revisited', *JOURNAL OF EXPERIMENTAL PSYCHOLOGY LEARNING MEMORY AND COGNITION* **18**, 1155–1155.

McNamara, T. P. and Altarriba, J. (1988), 'Depth of spreading activation revisited: Semantic mediated priming occurs in lexical decisions', *Journal of Memory and Language* **27**(5), 545–559.

McPherson, M., Smith-Lovin, L. and Cook, J. (2001), 'Birds of a feather: Homophily in social networks', *Annual review of sociology* pp. 415–444.

Merikle, P. M. and Reingold, E. M. (1991), 'Comparing direct (explicit) and indirect (implicit) measures to study unconscious memory', *Journal of Experimental Psychology: Learning, Memory, and Cognition* **17**(2), 224–233.

Meyer, D. E., Schvaneveldt, R. W. et al. (1971), 'Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations', *Journal of experimental psychology* **90**(2), 227–234.

Miller, G. A. (1995), 'Wordnet: a lexical database for english', *Communications of the ACM* **38**(11), 39–41.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D. and Miller, K. J. (1990), 'Introduction to wordnet: An on-line lexical database*', *International journal of lexicography* **3**(4), 235–244.

Nakov, P., Popova, A. and Mateev, P. (2001), Weight functions impact on lsa performance, *in* 'Proceedings of the EuroConference Recent Advances in Natural Language Processing, RANLP', Vol. 1, pp. 187–193.

Nelson, D. L., McEvoy, C. L. and Schreiber, T. A. (2004), 'The university of south florida free association, rhyme, and word fragment norms', *Behavior Research Methods, Instruments, & Computers* **36**(3), 402–407.

Nock, M. K., Park, J. M., Finn, C. T., Deliberto, T. L., Dour, H. J. and Banaji, M. R. (2010), 'Measuring the suicidal mind implicit cognition predicts suicidal behavior', *Psychological Science* **21**(4), 511–517.

Nosek, B. A., Banaji, M. R. and Greenwald, A. G. (2002), 'Harvesting implicit group attitudes and beliefs from a demonstration web site', *Group dynamics* **6**, 101–115.

Nosek, B. A., Greenwald, A. G. and Banaji, M. R. (2007), 'The implicit association test at age 7: A methodological and conceptual review', *Automatic processes in social thinking and behavior* pp. 265–292.

Padó, S. and Lapata, M. (2003), Constructing semantic space models from parsed corpora, *in* 'Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1', Association for Computational Linguistics, pp. 128–135.

Palermo, D. S. and Jenkins, J. J. (1964), 'Word association norms: Grade school through college.'.

Pennebaker, J. and King, L. (1999), 'Linguistic styles: language use as an individual difference.', *Journal of personality and social psychology* **77**(6), 1296.

Pennebaker, J., Mehl, M. and Niederhoffer, K. (2003), 'Psychological aspects of natural language use: Our words, our selves', *Annual review of psychology* **54**(1), 547–577.

Perkins, J. (2010), *Python text processing with NLTK 2.0 cookbook*, Packt Pub Limited.

Quillian, M. R. (1967), 'Word concepts: A theory and simulation of some basic semantic capabilities', *Behavioral science* **12**(5), 410–430.

Ratcliff, R., McKoon, G. et al. (1988), 'A retrieval theory of priming in memory', *Psychological review* **95**(3), 385–408.

Ratcliff, R., McKoon, G. et al. (1994), 'Retrieving information from memory: Spreading-activation theories versus compound-cue theories', *PSYCHOLOGICAL REVIEW-NEW YORK-* **101**, 177–177.

Ricoeur, P. (1976), *Interpretation theory: Discourse and the surplus of meaning*, Texas Christian University Press.

Rudman, L. A., Greenwald, A. G. and McGhee, D. E. (2001), 'Implicit self-concept and evaluative implicit gender stereotypes: Self and ingroup share desirable traits', *Personality and Social Psychology Bulletin* **27**(9), 1164–1178.

Rudman, L. A., Greenwald, A. G., Mellott, D. S. and Schwartz, J. L. (1999), 'Measuring the automatic components of prejudice: Flexibility and generality of the implicit association test', *Social Cognition* **17**(4), 437–465.

Simpson, J. A., Weiner, E. S. et al. (1989), *The Oxford english dictionary*, Vol. 2, Clarendon Press Oxford.

Sims, R. and Brinkmann, J. (2003), 'Enron ethics (or: culture matters more than codes)', *Journal of Business ethics* **45**(3), 243–256.

Wittgenstein, L. (1958), *Philosophical investigations*, Wiley-Blackwell.

Zipf, G. (1949), 'Human behavior and the principle of least effort.'.

Appendix A

Experiment materials

A.1 Graded Priming Stimulus

Target	**Associated Word**	**High T-Prime**	**Low T-Prime**
baby	child	hospital	room
kid	child	young	father
wave	brain	heat	radio
floor	ceiling	convention	manufacturer
town	city	resident	flame
nurse	doctor	army	public
ground	earth	earthquake	stake
girl	boy	death	love
truck	car	fire	sound
nation	country	newspaper	conscience
mind	memory	doubt	image
grass	green	acre	plane
hand	finger	cash	guard
wound	heal	bullet	blood
home	house	vacation	morning
woman	man	police	affair
letter	number	call	protest
game	play	war	season
sleep	bed	hour	day
food	stomach	emergency	flower
water	ocean	air	hole
window	door	bedroom	rain
law	justice	state	welfare

tree	leaf	family	branch
star	moon	movie	female
song	music	theme	show
crowd	people	cheer	candidate
ship	sea	passenger	transport
health	sick	public	package
army	soldier	officer	protest
smoke	tobacco	black	passenger

Table A.1: Graded Priming Stimulus: All stimulus words used by McKoon et al. (1992) that occurred more than 50 times in the Enron Corpus

A.2 Mediated Priming Stimulus

Target	**Prime**	**Mediated Prime**
beach	sand	box
war	peace	quiet
birthday	cake	pie
deer	animal	vegetable
eye	nose	smell
minute	hour	glass
soap	water	drink
priest	church	bell
ceiling	floor	carpet
hand	foot	kick
bat	ball	bounce
sky	blue	color
hard	soft	cotton
tea	coffee	bean
phone	number	letter
nurse	doctor	lawyer
reality	fantasy	island
blade	gun	trigger
circle	square	dance
mat	mouse	cheese
summer	winter	snow

wedding	ring	finger
teeth	brush	hair
sport	baseball	glove
rough	smooth	soft
cry	baby	bottle
bull	cow	milk
pen	pencil	lead
day	night	dark
white	black	coal
navy	army	tank
pretty	ugly	duck
moon	sun	hot
window	door	handle
school	bus	stop
valley	mountain	peak

Table A.2: Mediated Priming Stimulus: All stimulus words used by Balota and Lorch (1986) that occurred more than 50 times in the Enron Corpus

Appendix B

Code

B.1 database.py

```
##############################################################################################
# This file database.py contains all of the functions used in the preprocessing of the #
# enron corpus in preparation for semantic analysis                                          #
##############################################################################################

import pymysql as mdb
from pymysql.cursors import SSCursor
import sys,string
import math
import re
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.tag import BigramTagger, TrigramTagger, UnigramTagger
from nltk.stem import WordNetLemmatizer
import pickle
from nltk import FreqDist


##############################################################################################
# Function:connect_to_database():                                                            #
# Input: host of MySQL server,user account on server, password and database name             #
# Output: MySQLdb connection object                                                          #
##############################################################################################

def connect_to_database(host,user,password,dbname):
    try:
        con = mdb.connect(host,user,password,dbname)
        return con

    except dbmodule.Error, e:
        print "Error %d: %s" % (e.args[0],e.args[1])
        sys.exit(1)


##############################################################################################################
# Function: remove_emails()                                                                                  #
# Input: host of MySQL server,user account on server, password and database name                             #
# Output: In the new version of the database the following emails are removed: Emails sent outside           #
# 1999-2002; emails sent by users without @enron.com email addresses, emails sent by userid=256.            #
##############################################################################################################

def remove_emails(olddb,newdb,host,user,password):
```

```
con=connect_to_database(host,user,password,olddb)
con2=connect_to_database(host,user,password,newdb)
cur=con.cursor(SSCursor)
cur2=con2.cursor(SSCursor)
# remove emails outside 1999-2002
print "Removing emails outside 1999-2002"
for yr in range(1980,1999):
    cur.execute("SELECT messageid from messages WHERE YEAR(messagedt)=%d"%(yr))
    for messageid in cur:
        cur2.execute("DELETE FROM bodies where messageid=%d"%(messageid[0]))
        cur2.execute("DELETE FROM messages where messageid=%d"%(messageid[0]))
# remove emails from non @enron.com users
print "Removing from non @enron.com users"
cur.execute("SELECT personid,email From people WHERE enron=0")
for person in cur:
    cur2.execute("SELECT messageid from messages where senderid=%d"%person[0])
    answer=cur2.fetchall()
    for messageid in answer:
        cur2.execute("DELETE FROM bodies where messageid=%d"%(messageid[0]))
        cur2.execute("DELETE FROM messages where messageid=%d"%(messageid[0]))
# remove emails from user 256
print "Removing emails from user 256"
cur.execute("SELECT personid From people WHERE personid=256")
for person in cur:
    cur2.execute("SELECT messageid from messages where senderid=%d"%person[0])
    answer=cur2.fetchall()
    for messageid in answer:
        cur2.execute("DELETE FROM bodies where messageid=%d"%(messageid[0]))
        cur2.execute("DELETE FROM messages where messageid=%d"%(messageid[0]))
# remove people who have not sent any emails in the new version of the database
print "Removing people with no emails"
cur.execute("SELECT personid from people")
people=cur.fetchall()
for person in people:
    cur.execute("SELECT messageid from messages WHERE senderid=%d"%person[0])
    if len(cur.fetchall())==0:
        cur2.execute("DELETE FROM people where personid=%d"%(person[0]))
```

```
##########################################################################################################################
# Function: refine_content()                                                                                             #
# Input: old version of database,new version of database to record changes to, host of MySQL server,                     #
#        user account on server, password and database name.                                                             #
# Output: Forwarded messages,htmls, urls, email newline characters,email continuation characters, times and dates       #
```

```
#          are removed from each in email in the old version of the database and the changes written into the new  #
#          version of the database.                                                                                #
##################################################################################################################

def refine_content(olddb,newdb,host,user,password):
    con=connect_to_database(host,user,password,olddb)
    con2=connect_to_database(host,user,password,newdb)
    cur=con.cursor(SSCursor)
    cur2=con2.cursor(SSCursor)
    cur.execute("SELECT * from bodies")
    i=20
    for email in cur:
        # removes forwarded message
        match1 = re.search(".*\s*[Ff]orwarded\sby(.*\n)*",email[1])
        body = re.sub(".*\s*[Ff]orwarded\sby(.*\n)*","",email[1])
        # removes oringinal message
        match2 = re.search(".*[Oo]riginal\s[Mm]essage(.*\n*)*",body)
        body = re.sub(".*[Oo]riginal\s[Mm]essage(.*\n*)*","", body)
        # removes email adresses new table
        match3 = re.search("\S+@\S+\.\S+",body)
        body = re.sub("\S+@\S+\.\S+","", body)
        # remove url
        match4 = re.search("(https?://|www.|ftp.)+\S*",body)
        body = re.sub("(https?://|www.|ftp.)+\S*","", body)
        # remove email newline characters
        match5 = re.search("=20",body)
        body=re.sub("=20","",body)
        # remove character that represent a word carrying onto a new line
        match6 = re.search("=20",body)
        body=re.sub("=\\r\\n","",body)
        # remove dates
        match7 = re.search("\d?\d/\d\d/(\d\d)?(\d\d)?",body)
        body=re.sub("\d?\d/\d?\d/(\d\d)?(\d\d)?","",body)
        # remove times
        match8 = re.search("\d\d?:\d\d?(:\d\d)?",body)
        body=re.sub("\d\d?:\d\d(:\d\d)?","",body)
        # update new database version if change has been made
        if match1 or match2 or match3 or match4 or match5 or match6 or match7 or match8:
            body = mdb.escape_string(body)
            cur2.execute("UPDATE bodies SET body='%s' WHERE messageid=%d"%(body,email[0]))
        print "Reading email %d"%(email[0])


##################################################################################################################
```

```
# Function: token_and_lemmatize()                                                                                    #
# Input: old version of database,new version of database to record changes to, host of MySQL server, user account#
#        on server, password and database name.                                                                      #
# Output: Emails in new version of database are tokenised, lemmatised and punctuation is removed.                    #
#########################################################################################################################

def token_and_lemmatise(olddb,newdb,host,user,password):
    f = open('backofftagger.pickle', 'r')
    #get lemmatizer
    lemmatiser = WordNetLemmatizer()
    #get backoff POS tagger
    tagger = pickle.load(f)
    con=connect_to_database(host,user,password,olddb)
    con2=connect_to_database(host,user,password,newdb)
    cur=con.cursor(curse.SSCursor)
    cur2=con2.cursor(curse.SSCursor)
    cur.execute("SELECT * from bodies")
    for email in cur:
        #tokenise email
        sentences=tokenise_sentences(email[1])
        text=""
        for sentence in sentences:
            # covert words into lower case
            lower = map(lambda x: x.lower(),sentence)
            # pos tag words
            tagwords = tagger.tag(lower)
            # lemmatise postagged words
            mapwords = map(lambda x: lemma(x,lemmatiser),tagwords)
            # remove punctuation
            puntfree = map(lambda x:removepunc(x),mapwords)
            text+=mdb.escape_string(" ".join(puntfree))
        # write changes to new databae
        cur2.execute("UPDATE bodies SET body='%s' WHERE messageid=%d"%(text,email[0]))
        print "Read email %d"%(email[0])


#####################################################################
# Function: removepunc()                                            #
# Input: word token                                                 #
# Output:returns empty string if punctuation mark, word if word #
#####################################################################

def removepunc(word):
    if not re.search(".*[0-9A-Za-z]+.*",word):
        word=""
```

```
    if word!="n't":
        word = re.sub("['.]","",word)
    return word


###################################
#Function: tokenise_sentences() #
#Input: Text                     #
#Ouput: List of word tokens      #
###################################

def tokenise_sentences(text):
    sentences=sent_tokenize(text)
    return map(lambda x:word_tokenize(x),sentences)


#######################################
#Function: lemma()                    #
#Input: tagged word, lemmatiser       #
#Output: root lemma of tagged word #
#######################################

def lemma(tagword,lemmatiser):
    if tagword[1].startswith('N'):
        return lemmatiser.lemmatize(tagword[0],'n')
    if tagword[1].startswith('V'):
        return lemmatiser.lemmatize(tagword[0],'v')
    if tagword[1]=="ADJ":
        return lemmatiser.lemmatize(tagword[0],'a')
    if tagword[1]=="ADV":
        return lemmatiser.lemmatize(tagword[0],'r')
    else:
        return lemmatiser.lemmatize(tagword[0])


#########################################################################################################################
# Function: remove_small_emails()                                                                                       #
# Input: old version of database,new version of database to record changes to, host of MySQL server, user account#
#        on server, password and database name.                                                                         #
# Output: new version datbase is updated with emails that contain less than 10 words removed.                          #
#########################################################################################################################
```

```
def remove_small_emails(olddb,newdb,host,user,password):
    con=connect_to_database(host,user,password,olddb)
    con2=connect_to_database(host,user,password,newdb)
    cur=con.cursor(SSCursor)
    cur2=con2.cursor(SSCursor)

    # remove emails with less than 10 word tokens
    cur.execute("SELECT * from bodies")
    for email in cur:
        words = string.split(email[1])
        if len(words)<10:
            cur2.execute("DELETE FROM bodies where messageid=%d"%(email[0]))
            cur2.execute("DELETE FROM messages where messageid=%d"%(email[0]))
            print "Reading email %d"%(email[0])

    # remove people with no emails
    cur.execute("SELECT personid from people")
    people=cur.fetchall()
    for person in people:
        cur.execute("SELECT messageid from messages WHERE senderid=%d"%person[0])
        if len(cur.fetchall())==0:
            cur2.execute("DELETE FROM people where personid=%d"%(person[0]))


##########################################################################################################
# Function: annotate_negative()                                                                          #
# Input: old version of database,new version of database to record changes to, host of MySQL server, user account#
#        on server, password and database name.                                                          #
# Output: negative uses of words are tagged with the prefix 'not-' in the new version of the database   #
##########################################################################################################

def annotate_negative(olddb,newdb,host,user,password):
    con=connect_to_database(host,user,password,olddb)
    con2=connect_to_database(host,user,password,newdb)
    cur=con.cursor(SSCursor)
    cur2=con2.cursor(SSCursor)
    from nltk.corpus import stopwords as stop
    stopwords = stop.words("english")
    cur.execute("SELECT * from bodies")
    emailnum=0
    neg = 0
    for email in cur:
        emailnum+=1
        print "Reading email %d" %(emailnum)
```

```
            text = string.split(email[1])
            length =len(text)
            for i in range(length):
                # find negative token
                if text[i] == "n't" or text[i] == "not":
                    j = i+1
                    while j<length and text[j] in stopwords:
                        j+=1
                    if j<length:
                        # annotate negative use of adverb,nouns, verb, or adjective
                        text[j]="not-"+text[j]
                    i=j
                    neg+=1
            text2=mdb.escape_string(" ".join(text))
            cur2.execute("UPDATE bodies SET body='%s' WHERE messageid=%d"%(text2,email[0]))
        print "Found %d negative words" % (neg)


#############################################################################################################################
# Function: preprocess()                                                                                                    #
# Input: null                                                                                                               #
# Output: All 5 preprocessing stages are performed, with each change updated in a new version of the enron corpus#
#############################################################################################################################

def preprocess():
    # MySQL settings:
    host="localhost"
    user="guest"
    password="corpus"

    #Stage 1:Delete emails
    #remove_emails("enronoriginal","enronprocessed1",host,user,password)

    #Stage 2:Refine emails
    #refine_content("enronprocessed1","enronprocessed2",host,user,password)

    #Stage 3:Tokenise and lemmatise corpus
    # token_and_lemmatise("enronprocessed2","enronprocessed3",host,user,password)

    #Stage 4:Remove small emails from the corpus
    #remove_small_emails("enronprocessed3","enronprocessed4",host,user,password)

    #Stage 5:Annotate negative uses of words
    annotate_negative("enronprocessed4","enronfinal",host,user,password)
```

```
############################
# Run database.py as main #
############################
if __name__=="__main__":
    preprocess()
```

B.2 postagger.py

```
#############################################################################
# This file postagger.py provides functions that allow a backoff postagger#
# to be created for use in database.py when lemmatising the Enron Corpus  #
#############################################################################
import pickle
from nltk.tag import BigramTagger, TrigramTagger, UnigramTagger, DefaultTagger
from nltk.corpus import brown


#############################################################################################
# Function: backoff_tagger()                                                                #
# Input: tagged sentences as training data, list of tagger classes, default backoff tagger#
# Output: trained backoff tagger                                                            #
#############################################################################################

def backoff_tagger(trainingdata, tagger_classes, backoff=None):
    # for each class in tagger_classes create a tagger object with the backoff tagger specified
    for cls in tagger_classes:
        backoff = cls(trainingdata, backoff=backoff)
    return backoff


#############################################################################
# Function: create_tagger()                                                 #
# Input: Null                                                               #
# Output: Creates a backoff POS tagger from using the Brown Corpus as training data#
#############################################################################

def create_tagger():
    f = open('backofftagger.pickle', 'w')
    # Get Brown corpus tagged sentences
    trainingdata = brown.tagged_sents(simplify_tags=True)
```

```
    taggers = []
    tagger = backoff_tagger(trainingdata, [UnigramTagger, BigramTagger, TrigramTagger], DefaultTagger('NN'))
    print "Accuracy of tagger is %f" % (tagger.evaluate(trainingdata))
    pickle.dump(tagger, f)
    f.close()


##############################
# Run postagger.py as main#
##############################
if __name__ == "__main__":
    create_tagger()
```

B.3 words.py

```
#########################################################################################################################
# file words.py is used to anaylse the token distribution in the enron corpus, obtain a set of target words#
# and obtain differently sorted context words                                                                           #
#########################################################################################################################
import MySQLdb as mdb
import MySQLdb.cursors as curse
import math
import re
import cPickle as pickle
from nltk import FreqDist
from database import connect_to_database
from nltk.corpus import names as name
from nltk.corpus import wordnet as wordnet
from nltk.corpus import stopwords as stop
import string
from scipy import stats
from semanticspace import *
from numpy import var
import numpy as np
import enchant


#############################################################################
# Function list_to_dic()                                                    #
# Input: List                                                               #
# Output: Dictionary where each item in the list is a key with the value of 1#
```

```
#############################################################################

def list_to_dic(lst):
    dic={}
    for i in lst:
        dic[i.lower()]=1
    return dic


#############################################################################################
# Function: word_stats()                                                                    #
# Input: Null                                                                               #
# Output: CSV file containing the proportion of types of word tokens, CSV of the top 10 words#
#          ,CSV of the top 10 words excluding stopwords and CSV of top 10 unrecognised words  #
#############################################################################################

def word_stats():
    nonwords=0
    words=0
    namewords=0
    unrecwords=0
    totalwords=0

    wordcount=0
    nonstopcount=0
    unreccount=0

    malenames= list_to_dic(name.words("male.txt"))
    femalenames= list_to_dic(name.words("female.txt"))
    d1 = enchant.Dict("en_US")
    d2 = enchant.Dict("en_UK")
    stopwords = stop.words('english')

    # create frequency distribution dictionary
    fd=FreqDist()
    output1 = open("Results/tokenproportions.csv","w")
    output2 = open("Results/top10words.csv","w")
    output3 = open("Results/top10nonstopwords.csv","w")
    output4 = open("Results/top10unrecwords.csv","w")

    con=connect_to_database("localhost","guest","corpus","enronfinal")
    cur=con.cursor(curse.SSCursor)
    cur.execute("SELECT body from bodies")
```

```
    # calculate proportion of types of words in the corpus
    for email in cur:
        text = string.split(email[0])
        for token in text:
            totalwords+=1
            # Load words into frequecy distribution dictionary
            fd.inc(token)

            if d1.check(token) or d2.check(token) or wordnet.synsets(token):
                words+=1
            elif token in malenames or token in femalenames:
                namewords+=1
            else:
                if not re.match("^[a-z]+$",token):
                    nonwords+=1
                else:
                    unrecwords+=1

    # output proportion to CSV file
    output1.write("Words not contained in the English American or WordNet dictionaries,%f%%,\n"
        %(float(words)*100/totalwords))
    output1.write ("Unrecognised words,%f%%,\n" %(float(unrecwords)*100/totalwords))
    output1.write("Non words,%f%%,\n" %(float(nonwords)*100/totalwords))
    output1.write("Male and female names,%f%%\n" %(float(namewords)*100/totalwords))
    print totalwords

    # calculate the 10 most frequent words, words excluding stop words and unrelated words
    for token in fd:
        if wordcount==10 and nonstopcount == 10 and unreccount == 10:
            return
        if d1.check(token) or d2.check(token) or wordnet.synsets(token):
            if re.match("^[A-Za-z]+([A-Za-z]|-)*$",token) and token not in stopwords and nonstopcount<10 and
                token!="would":
                output3.write("%s,%d,\n"%(token,fd[token]))
                nonstopcount+=1
            elif wordcount<10:
                output2.write("%s,%d,\n"%(token,fd[token]))
                wordcount+=1
        elif re.match("^[a-z]+$",token) and unreccount<10:
            output4.write("%s,%d,\n"%(token,fd[token]))
            unrecwords+=1

#print "word stats"
#word_stats()
```

```
#############################################################
# Function: get_targetwords()                               #
# Input: Null                                               #
# Output: Set of target words from the Enron corpus         #
#############################################################
def get_targetwords():
    output=open("Wordfiles/targetwords2.txt","w")
    pronouns = "he her him herself him himself his i me my myself our ourselves she their themselves them they us
        we you your yourself "
    companies = "enron dynergy"
    con=connect_to_database("localhost","guest","corpus","enronfinal")
    cur=con.cursor(curse.SSCursor)
    cur.execute("SELECT body from bodies")
    # create frequency distribution dictionary
    fd = FreqDist()
    stopwords = stop.words('english')
    d1 = enchant.Dict("en_US")
    d2 = enchant.Dict("en_UK")
    for email in cur:
        text = string.split(email[0])
        for token in text:
            if (d1.check(token) or d2.check(token) or wordnet.synsets(token)) and not token in stopwords and
                re.match("^[A-Za-z]+([A-Za-z]|-)*$",token) and token!="would":
                # increment count for word in frequency distribution
                fd.inc(token)

    # add all target words that occur more than 50 times
    for token in fd:
        count = fd[token]
        if count<50:
            break
        else:
            output.write(token+ " ")
    # also include pronouns and companies
    output.write(pronouns)
    output.write(companies)

print "target words"
get_targetwords()
print "conent words"
get_contentwords()

#############################################################################################
```

```
# Function: get_freq_contentwords()                                                    #
# Input: Null                                                                          #
# Output:Text file of content words ordered by frequency from the Enron corpus         #
#######################################################################################

def get_freq_contentwords():
    output=open("Wordfiles/frequent-basis2.txt","w")
    con=connect_to_database("localhost","guest","corpus","enronfinal")
    cur=con.cursor(curse.SSCursor)
    cur.execute("SELECT body from bodies")
    # create frequency distribution dictionary
    fd = FreqDist()
    stopwords = stop.words('english')
    d1 = enchant.Dict("en_US")
    d2 = enchant.Dict("en_UK")
    for email in cur:
        text = string.split(email[0])
        for token in text:
            if (d1.check(token) or d2.check(token) or wordnet.synsets(token)) and not token in stopwords and
                re.match("^[A-Za-z]+([A-Za-z]|-)*$",token) and token!="would":
                # increment count for word in frequency distribution
                fd.inc(token)

    # add all target words that occur more than 50 times
    for token in fd:
        count = fd[token]
        if count<50:
            break
        else:
            output.write(token+ " ")



#######################################################################################################
# Function: get_reliable()                                                                            #
# Input: Four Semantic spaces, each analysed from a different section of the corpus (see              #
#        commented out section in SemanticSpace.movewindow() in semanticspace.py )                    #
# Output: Text file of content words ordered by reliability from the Enron Corpus                     #
#######################################################################################################

def get_reliable(space1,space2,space3,space4):
    spaces=[space1,space2,space3,space4]
    pvals={}
    fvals={}
    freqs=[]
```

```
    # create frequency distribution
    fdist = FreqDist()
    anovatxt=open("Wordfiles/reliable-basis2.txt","w")

    fout=open("fvals.csv","w")
    pout=open("pvals.csv","w")
    contextindex=spaces[0].columnindex
    # calculate the reliability of each word
    for context in contextindex:
        tempcolumns=[]
        totalfreq=0
        # perform an anova test over the 4 columns of the context word
        for space in spaces:
            tempcolumns.append(space.get_column(context))
            totalfreq+=space.contextfreq.get(context,0)
        f_val, p_val = stats.f_oneway(*tempcolumns)
        # multiply the p_val by 10^30 and turn into integer
        val = p_val * pow(10,30)
        if val<=0:
            val=1
        try:
            val = int(val)
        except ValueError:
            val=1
        # store the conext word and its p_val in the frequency distribution
        fdist.inc(context,count=val)

    # output frequency distribution sorted by p_vals into a text file
    for key in fdist:
        anovatxt.write(key+"\n")
    fout.close()
    pout.close()


##############################################################################################
# Function: get_reliable()                                                                   #
# Input: Single Semantic space of word use in the Enron corpus                               #
# Output: Text file of content words ordered by variablity from the Enron Corpus             #
##############################################################################################

def variance(space):
    # get frequency distriubtion
    fd=FreqDist()
    output = open("Wordfiles/variable-basis2.txt","w")
    length = len(space.columnindex)
```

```
    i=0
    # calculate the variance of each context column
    for context in space.columnindex:
        print "Reading %d of %d" % (i,length)
        column=space.get_column(context)
        # multiply variance by 10^30, convert to an integer, add context word and
        # variance to frequency distribution
        variance=var(column)*pow(10,30)
        fd.inc(context,count=int(variance))
        i+=1
    # output frequency distribution sorted by variability into a text file
    for context in fd:
        output.write(context +" ")
```

B.4 semanticspace.py

0.75

```
#################################################################################################
# file semanticspace.py contains the class from creating a sematic space of the enron corpus    #
#################################################################################################

import pymysql as mdb
from pymysql.cursors import SSCursor
import string
import math
import cPickle as pickle
from scipy import linalg, mat, dot,sparse
from sys import exit
from numpy import var
import numpy as np

#########################################################################################################################
# Class: SemanticSpace                                                                                                  #
# Initial Parameters: target word filename, context word filename, number of context words                              #
#                     in the target file to use where 0 refers to using all context words                               #
# Description: Class for creating a semantic space of the Enron corpus. It contains the following public functions:    #
#              window_analysis(), log_odds(), normalise(), svd(), get_column(), get_sim() and semantic_analysis         #
#########################################################################################################################
class SemanticSpace:
    def __init__(self,tfilename,cfilename,cnumber):
        # self.weigted: 0 for Not Weighted, 1 for Weighted
        self.weighted=0
```

```
        # window size used in obtaining semantic space
        self.window_size=0
        # total number of word tokens in the semantic space
        self.wordnumber=0
        # dictionary of target word frequencies
        self.targetfreq={}
        # dictionary of context word frequencies
        self.contextfreq={}
        # lexical function used in obtaining semantic space
        self.lexicalfun="None"
        # used to display progress of analysis in the GUI
        self.outputprint="␣"
        # name of target word file
        self.targetname=tfilename
        # name of context word file
        self.contextname=cfilename
        # hashtable of target word:coocur row index
        self.rowindex=self.__get_index(tfilename,0)
        # hashtable of context word: coocur column index
        self.columnindex=self.__get_index(cfilename,cnumber)
        # co-occurrence matrix
        self.coocur=self.__get_init_matrix(self.rowindex,self.columnindex)
        #number of dimensions used in SVD
        self.dimensions=len(self.columnindex)


#########################################################################################################
#  Function: widow_analysis()                                                                           #
#  Input:    Window size,type of window (1 for weighted,0 for not weighted),database connection         #
#  Output:   Co-occurrence matrix is populated with targetword-contextword co-occurrences using       #
#            the window passing technique                                                               #
#########################################################################################################

def window_analysis(self,wsize,weighted,con):
    self.window_size=wsize
    self.weighted=weighted
    cur=con.cursor(SSCursor)

    #count number of emails
    cur.execute("SELECT␣COUNT(*)␣FROM␣bodies")
    totalemail=cur.fetchall()[0][0]

    # buffer all emails on sever side cursor
    cur.execute("SELECT␣body␣FROM␣bodies")
```

```
    # Commented out section:
    # Used to analyse 4 individual sections of the enron corpus, rather than the whole corpus (used in
        get_reliable() in words.py)
    # ------------------------------------------------------------------------------------------------
    # cur.execute("SELECT body FROM bodies where messageid<=70419") used to create semanticspacepart1
    # cur.execute("SELECT body FROM bodies where (messageid<=124599 AND messageid>70419)") used to create
        semanticspacepart2
    # cur.execute("SELECT body FROM bodies where (messageid<=167673 AND messageid>124599)") used to create
        semanticspacepart3
    # cur.execute("SELECT body FROM bodies where (messageid<=223574 AND messageid>167673)") used to create
        sematicspacepart4

    i=1
    # move a window through each email
    for email in cur:
        # update GUI output
        self.outputprint = "Reading email %d of %d using window size of %d" % (i,totalemail,wsize)
        self.__move_window(email[0],wsize,weighted)
        i+=1


##########################################################################################################
# Function: __get_index()                                                                                #
# Input:    contextword or targetword filename, number of words to use (0 is for all words)              #
# Output:   hashtable of either columnindexes for context words or row indexes for targetwords           #
##########################################################################################################

def __get_index(self,filename,number):
    file=open(filename,"r")
    index=0
    indexdic={}
    j=0
    line = file.readline()
    while line:
        wordlist=string.split(line)
        wordlist = map(lambda x:x.lower(),wordlist)
        for word in wordlist:
            j+=1
            if not indexdic.has_key(word):
                indexdic[word] = index
                index+=1
            if j==number:
                return indexdic
```

```
            line = file.readline()
        return indexdic


#####################################################################
# Function: __get_init()                                            #
# Input: row hashtable, column hashtable                            #
# Output: Co-occurence matrix initialised with 0s  in all cells#
#####################################################################

    def __get_init_matrix(self,rowindex,columnindex):
        cooccur=[]
        for i in range(0,len(rowindex)):
            cooccur.append([])
            for j in range(0,len(columnindex)):
                cooccur[i].append(0)
        return cooccur


#############################################################################################
# Function: __move_window()                                                                 #
# Input: email body,window size,type of window                                              #
# Output: window is passed through the corpus and the co-occurence matrix is updated#
#############################################################################################

    def __move_window(self,text,windowsize,weighted):
        # initialise window
        window=self.__init_window(windowsize)
        words=string.split(text)
        # pass window up until end of email a word at a time
        for word in words:
            # update word frequecy
            self.__update_total_freq(word)
            self.wordnumber+=1
            window.pop(0)
            window.append(word)
            # analyse current state of the window
            self.__analyse_window(window,weighted,windowsize)
        # moves the window past the end of the email
        while window[0]!="":
            window.pop(0)
            window.append("")
            # analyse current state of window
            self.__analyse_window(window,weighted,windowsize)
```

```
##########################################################################################################
# Function: __update_total_freq()                                                                        #
# Input: word                                                                                            #
# Output: word frequency dictionaries are updated if word is a target word or context word              #
##########################################################################################################

def __update_total_freq(self,word):
    if self.columnindex.has_key(word):
        self.contextfreq[word]=self.contextfreq.get(word,0)+1
    if self.rowindex.has_key(word):
        self.targetfreq[word]=self.targetfreq.get(word,0)+1


##########################################################################################################
# Function: __init_window()                                                                              #
# Input: Window size                                                                                     #
# Output: Queue of size = window size, intitalised with empty strings in all cells                       #
##########################################################################################################

def __init_window(self,windowsize):
    lst = []
    for i in range(windowsize):
        lst.append("")
    return lst


##########################################################################################################
# Function: analyse_Window()                                                                             #
# Input:    Window of words, type of window, window size                                                 #
# Output:   Updates co-occurence values for head and tail of window                                      #
##########################################################################################################
def __analyse_window(self,window,weighted,windowsize):
    firstword=window[0]
    lastword=window[windowsize-1]
    targetfirst=self.rowindex.has_key(firstword)
    targetlast=self.rowindex.has_key(lastword)

    if targetfirst or targetlast:
        firstvalue=1
        lastvalue=1
```

```
        for i in range(len(window)):
            windowword=window[i]
            wordiscontext=self.columnindex.has_key(window[i])
            if wordiscontext:
                # if weighted change increment ammount
                if weighted:
                    firstvalue=windowsize-i
                    lastvalue=i+1
                if targetfirst and i!=0:
                    self.__update_cooccur(firstword,windowword,firstvalue)
                if targetlast and i!=len(window)-1:
                    self.__update_cooccur(lastword,windowword,lastvalue)


##############################################################################
# Function : __update_coocur                                                 #
# Input: target word, context word, value to increment co-occurence count by #
# Output: Co-occurences are updated by the value for target word-contecxt word pair#
##############################################################################

def __update_cooccur(self,target,context,value):
    # get row and column index of pair and update co-occurence count
    row=self.rowindex[target]
    column=self.columnindex[context]
    self.coocur[row][column]+=value


##########################################
# Function: __freq_t()                   #
# Input: target word                     #
# Output: freqency count for target word #
##########################################

def __freq_t(self,target):
    return self.targetfreq.get(target,0)


##########################################
# Function: __freq_c()                   #
# Input: context word                    #
# Output: freqency count for context word#
##########################################
```

```
def __freq_c(self,context):
    return self.contextfreq.get(context,0)


##############################################################################
# Function: __freq_t_c()                                                     #
# Input: target word, context word                                           #
# Output: frequency of co-occurence of the target word with the context word #
##############################################################################

def __freq_t_c(self,target,context):
    if self.rowindex.has_key(target) and self.columnindex.has_key(context):
        return self.coocur[self.rowindex[target]][self.columnindex[context]]
    else:
        return 0


##############################################################################
# Function: __freq_t_nc()                                                    #
# Input: target word, context word                                           #
# Output: Number of times target word occurs without the context word        #
##############################################################################

def __freq_t_nc(self,target,context,windowsize):
    return (windowsize*self.__freq_t(target)) - self.__freq_t_c(target,context)


##############################################################################
# Function: __freq_c_nt()                                                    #
# Input: target word, context word                                           #
# Output: Number of times context word occurs without the target word        #
##############################################################################

def __freq_c_nt(self,target, context,windowsize):
    return (windowsize*self.__freq_c(context)) - self.__freq_t_c(target,context)

##############################################################################
# Function: __freq_c_nt()                                                    #
# Input: target word, context word                                           #
# Output: Number of times neither target word or context word occur in the window#
##############################################################################

def __freq_nc_nt(self,target,context,windowsize):
```

```
    nw = self.wordnumber * windowsize
    return nw - self.__freq_c_nt(target,context,windowsize) - self.__freq_t_nc(target,context,windowsize) -
        self.__freq_t_c(target,context)


##############################################################################
# Function: log_odds()                                                       #
# Input: size of window                                                      #
# Output: performs the log odds ratio on the whole of the co-occurence matrix #
##############################################################################

def log_odds(self,windowsize):
    self.lexicalfun="Log Odds Ratio"
    # create temporary matrix
    logsmatrix=[]
    targetwords = self.rowindex.keys()
    totaltarget = len(targetwords)
    contextwords = self.columnindex.keys()
    windowsize=(windowsize*2)-1
    # copy cooccurence matrix into temporary matrix
    for target in targetwords:
        logsmatrix.insert(0,[])
        for context in contextwords:
            logsmatrix[0].append(0)
    i=1
    # perform log odds ratio on temporary matrix
    for target in targetwords:
        self.outputprint="Performing logs odds ratio on row %d of %d" % (i,totaltarget)
        i+=1
        for context in contextwords:
            ftc=self.__freq_t_c(target,context)
            # if co-occurence count of target word and context word is 0 then move on
            if(ftc==0):
                logsmatrix[self.rowindex[target]][self.columnindex[context]]=0
                continue
            fntnc=self.__freq_nc_nt(target,context,windowsize)
            ftnc=self.__freq_t_nc(target,context,windowsize)
            fcnt=self.__freq_c_nt(target,context,windowsize)
            # replace co-occurence count with log odds ratio
            logsmatrix[self.rowindex[target]][self.columnindex[context]]=math.log(float((ftc*fntnc))/(ftnc*fcnt))
            # if log odds ratio is negative then set to 0
            if logsmatrix[self.rowindex[target]][self.columnindex[context]] <0:
                logsmatrix[self.rowindex[target]][self.columnindex[context]]=0
    # copy log odds matrix into co-occurence matrix
    self.coocur=logsmatrix
```

```
##############################################################################################
# Function: log_odds()                                                                       #
# Input: Null                                                                                #
# Output: performs normalisation on co-occurence matrix                                      #
##############################################################################################
def normalise(self):
    self.lexicalfun="Normalisation"
    totalkeys = len(self.rowindex.keys())
    i=1
    for target in self.rowindex.keys():
        freq=self.freq_t(target)
        row=self.coocur[self.rowindex[target]]
        rowsum=sum(row)
        self.outputprint="Normalising row %d of %d" % (i,totalkeys)
        for j in range(len(row)):
            if rowsum!=0:
                # normalise cell
                self.coocur[self.rowindex[target]][j]=float(row[j])/rowsum
        i+=1


##############################################################################################
# Function: svd()                                                                            #
# Input: number of dimensions                                                                #
# Output: performs SVD and reduces co-occurence matrix by number of dimensions               #
##############################################################################################

def svd(self,dimensions):
    print "hello"
    self.dimensions=dimensions
    # convert matrix into numpy scipy matrix
    tempmatrix=mat(self.coocur,dtype=np.float32)
    self.outputprint= "Tranforming matrix into the form: U,s,V"
    print "hello2"
    # reduce matrix to U,s,V
    U,s,V = linalg.svd(tempmatrix,full_matrices=False)
    self.outputprint= "Reducing Dimensions to %d" %(600)
    print "hello3"
    # set all but specified dimensions to 0
    for index in xrange(dimensions, len(s)):
        s[index]=0
    # reconstruct matrix
    self.coocur=dot(dot(U,linalg.diagsvd(s,len(s),len(V))),V)
```

```
    # convert co-occurence matrix back into a list
    self.coocur=self.coocur.tolist()


#############################################################################
# Function: get_column()                                                    #
# Input: context word                                                       #
# Output: returns the column of the context word as a list                  #
#############################################################################

def get_column(self,context):
    column=[]
    for row in self.coocur:
        column.append(row[self.columnindex[context]])
    return column


#############################################################################
# Function: __cosine()                                                      #
# Input: two row vectors to compare                                         #
# Output: the cosine angle between the two row vectors                      #
#############################################################################

def __cosine(self,vector1,vector2):
    normvector1=0.0
    normvector2=0.0
    dotproduct=0
    for i in range(0,len(vector1)):
        dotproduct+=(vector1[i]*vector2[i])
        normvector1+=math.pow(vector1[i],2)
        normvector2+=math.pow(vector2[i],2)
    if math.pow(normvector1,0.5)*math.pow(normvector2,0.5) == 0:
        return 0
    cosine=float(dotproduct)/(math.pow(normvector1,0.5)*math.pow(normvector2,0.5))
    return cosine


#############################################################################
# Function: __euclidean()                                                   #
# Input: two row vectors to compare                                         #
# Output: the euclidean distance between the two row vectors                #
#############################################################################
```

```
def __euclidean(self,vector1,vector2):
    distance=0.0
    for i in range(0,len(vector1)):
        distance+=pow((vector1[i]-vector2[i]),2)
    distance=pow(distance,0.5)
    return distance

##############################################################################
# Function: __cityblock()                                                    #
# Input: two row vectors to compare                                          #
# Output: the city block distance between the two row vectors                #
##############################################################################

def __cityblock(self,vector1,vector2):
    distance=0.0
    for i in range(0,len(vector1)):
        distance+=math.fabs(vector1[i]-vector2[i])
    return distance

##############################################################################
# Function: __hellinger()                                                    #
# Input: two row vectors to compare                                          #
# Output: the hellinger distance between the two row vectors                 #
##############################################################################

def hellinger(self,vector1,vector2):
    similarity=0.0
    for i in range(0,len(vector1)):
        similarity+=pow(pow(vector1[i],0.5)-pow(vector2[i],0.5),2)
    return similarity



##############################################################################
# Function: __kullback()                                                     #
# Input: two row vectors to compare                                          #
# Output: the kullback distance between the two row vectors                  #
##############################################################################

def kullback(self,vector1,vector2):
    similarity=0.0
    for i in range(0,len(vector1)):
        if vector2[i]!=0 and vector1[i]!=0:
            similarity+=vector1[i]*math.log(float(vector1[i])/vector2[i])
    return similarity
```

```
##############################################################################
# Function: get_sim()                                                        #
# Input:  two words and the similarity metric (Cosine, Euclidean, City Block #
#         Hellinger or Kullback Liebler                                      #
# Output: similarity value between the two words using the similarity metric #
#         specified                                                          #
##############################################################################

def get_sim(self,word1,word2,metric):
    if not word1 in self.rowindex or not word2 in self.rowindex:
        return 0
    else:
        vector1=self.coocur[self.rowindex[word1]]
        vector2=self.coocur[self.rowindex[word2]]
        if metric == "Cosine":
            return self.__cosine(vector1,vector2)
        elif metric == "Euclidean":
            return self.__euclidean(vector1,vector2)
        elif metric == "City_Block":
            return self.__cityblock(vector1,vector2)
        elif metric == "Hellinger":
            return self.__hellinger(vector1,vector2)
        elif metric == "Kullback-Liebler":
            return self.__kullback(vector1,vector2)


##########################################################################################################################
# Function: semantic_analysis                                                                                            #
# Input: Window size, MySQL connection, windowtype (1 for Weighted,0 for Not Weighted),                                  #
#        lexical function (either "Normailsation" or "Log_Odds_Ratio"), number of dimensions to perform SVD on           #
# Output: All steps: window passing, lexical association and dimensionality reduction are performed on the               #
#         semantic space                                                                                                 #
##########################################################################################################################

def semantic_analysis(self,windowsize,con,weighted,lexicalfun,dimensions):
    self.window_analysis(windowsize,weigted,con)
    if lexicalfun == "Normalisation":
        self.normalise()
    elif lexicalfun == "Log_Odds_Ratio":
        self.log_odds(windowsize)
    if dimensions!=0:
```

```
self.svd(dimensions)
```